Digital Twin using Multivariate Prediction

Jere Puolakanaho
2380155
Research Unit of Mathematical Sciences
University of Oulu

Spring 2019

Abstract

Digital Twin is a digital replica of physical assets, processes and systems that can be used for various purposes. Virtual model is constructed from the corresponding physical model and these two are then connected by generating real time data using sensors. Resent advances in technology have enabled this, because they have made digital environments cost-efficient. The goal is to simulate the real world with minimal physical resources using machine learning, multivariate analysis and other mathematical techniques. Optimally no physical resources go to waste in testing and development.

Runtime validation means monitoring and validating a running system with both digital and physical models. Simulated (digital) model parameters are used in the physical model and then the simulated model is updated with new data from the physical model. Simulated model gains more and more data over time becoming more accurate.

This thesis studies the applicability of mathematical models as a prediction tool to predict and validate systems behaviour as a part of simulation. And further, be used in analysis in a digital twin model.

Keywords: Digital Twin, multivariate analysis, prediction, runtime validation

Tiivistelmä

Digitaalinen kaksonen on digitaalinen kopio fyysisistä voimavaroista, prosesseista ja systeemeistä, jota voidaan käyttää moniin tarkoituksiin. Virtuaalinen malli rakennetaan vastaavasta fyysisestä mallista ja nämä yhdistetään toisiinsa luomalla reaaliaikaista dataa sensoreilla. Viimeaikaiset kehitykset teknologiassa ovat mahdollistaneet tämän tekemällä digitaalisista ympäristöistä kustannustehokkaita. Tehtävänä on simuloida oikeaa maailmaa minimaalisilla fyysisillä resursseilla käyttäen koneoppimista, monimuuttuja-analyysiä ja muita matemaattisia tekniikoita. Parhaimmillaan fyysisiä resursseja ei menisi hukkaan testauksessa ja kehityksessä.

Ajonaikainen validointi tarkoittaa toimivan systeemin tarkkailua ja validointia fyysisen ja digitaalisen mallin avulla. Simuloidun (digitaalisen) mallin arvoja käytetään fyysisessä mallissa ja simuloitua mallia päivitetään uudella datalla fyysisen mallin tulosten perusteella. Simuloitu malli saa yhä enemmän dataa ajan saatossa ja tätä myötä paranee.

Tämä tutkielma tutkii matemaattisten mallien käytettävyyttä ennustukseen ja systeemin validointiin sekä simulaatiossa että digitaalisen kaksosen analyysissä.

Avainsanat: Digitaalinen kaksonen, monimuuttuja-analyysi, ennustaminen, ajonaikainen validointi

Foreword

I want to thank Nokia Solutions and Networks for providing me this opportunity to do my master's thesis for their company. This process of making the thesis was a fascinating and educational journey. I have learned a lot about Nokia as a company and community. New topics that came through this journey have kindled my interest to continue on this career/profession.

From the Nokia Solutions and Networks, I want to especially thank my thesis supervisor Mr. Pekka Tuuttila, and Mrs. Kirsti Simula. Your technical guidance and feedback helped me a lot during this process. Thank you for keeping me motivated.

From the University of Oulu, I want to thank my principal supervisor docent Erkki Laitinen for guiding me in my studies and through this process. I also want to thank Leena Ruha, PhD, for being the second examiner of my thesis. I want to thank the University of Oulu and the Research Unit of Mathematical Sciences for offering magnificent education during these five years.

This master thesis work has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737494. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Sweden, France, Spain, Italy, Finland, Czech Republic.

Abbreviations and symbols

Acronyms

SS_{res} Sum of Squares of Residuals

SS_{tot} Total Sum of Squares

ALD Analysis-Led Design

ANN Artificial Neural Network

DTA Digital Twin Aggregate

DTE Digital Twin Environment

DTI Digital Twin Instance

DTP Digital Twin Prototype

LS-SVM Least-Squares Support-Vector Regression

MVA Multivariate Analysis

NIPALS Non-linear Iterative Partial Least Squares

PCA Principle Component Analysis

PCR Principal Component Regression

PLM Product Lifecycle Management

PLS Partial Least Squares or Projection to Latent Structures

PRESS Predictive Error Sum of Squares

RDA Redundancy Analysis

RMSEP Root Mean Squared Error of Prediction

SVD Singular Value Decomposition

SVM Support-Vector Machine

SVR Support-Vector Regression

VIP Variable Importance in Projection

Symbol

λ Eigenvalue

B Regression coefficient matrix

C Loading of Y

C_{xy} Covariance matrix of X and Y

E Error matrix

H Score of Y

i Index number

j Index number

P Loading of X

$P_w(x)$ The projection of a vector x onto a vector w

Q^2 Goodness of prediction

R^2 Goodness of fit

T Score of X

X Input matrix

x_{ij} Element of input matrix

Y Output matrix

y_{ij} Element of output matrix

Contents

1 Introduction

Creating and testing products or processes require a lot of resources and time. Just a prototype can cost a fortune and not even work properly. Further testing can be a laborious task and essentially lead to the unseen problems. These unseen problems are many, but most harming would be the incapable hardware. Meaning, new parts must be produced costing the company more money. The current product might perform well in the preliminary testing, but further testing shows faults and errors which again leads to additional costs. To avoid unnecessary work and to prevent errors beforehand, simulating has been considered as a probable solution. Simulation, we speak of, is virtually imitating a process or product with algorithms and mathematical models. Potential problems could be seen before production or after the first prototypes.

Virtual environments have become a necessity these days and for a reason. Digitalisation has enabled us to perform more complex tasks virtually and these virtual tasks are cheaper and faster than physical tasks. Think of an algorithm that can do a rather complex task in matter of seconds. Human doing the same would require hours or days and not even reach the same precision as a computer. Now imagine if the same could be done to an engine. Many moving parts and a huge number of variables to take into consideration. This can be done using simulation. If there is data regarding a process, it can be simulated virtually, cutting costs and time used significantly. After process is simulated, built and put to use, data can be collected and once again simulated. At this point those simulations can improve the process further. To be precise, past data is used to predict how a process behaves in new areas. As time goes by, data volume grows.

The size of the data has greatly increased as processes have become more complex. Univariate analysis has been the way of the world for quite a while now, but since the amount of variables have increased to hundreds, we encounter a problem. Traditional univariate analysis methods struggle analysing when there are more than a few variables. And even if variables are analysed one by one, there are still relations between variables that univariate analysis might not notice. Thus, multivariate analysis seems to be a solution. Multivariate methods are sufficient here, since now we are focusing on multiple variables and the relations between them.

Even though data is collected continuously and certain values are

monitored actively, most error analyses are done long after an error has occurred as a post-analysis. This can be disastrous in many cases as most of the equipment might be harmed. More adequate approach would be to have some initiative and prevent such event from happening beforehand. To achieve that, we consider predicting the process behaviour with some degree of accuracy. Simulation could give a hint of errors and reasons to them beforehand. Proper countermeasures could be made and resources saved.

Reasons are many to research simulation. As said before, a large amount of resources could be saved and redirected to other purposes and there is a good chance that one way of simulation could be implemented on various cases. If that is indeed the case and a proper simulation methods/algorithms are found, then cloudification of these methods would further enhance their usability. Multiple personnel of a company could run their cases regardless of the location or the programs they hold access to. As long as they have access to the cloud, simulation can be used. The usage of data and simulation bring efficiency to the project and the whole industry.

1.1 Scope of thesis

Thesis focuses on the simulation of a multivariate data via multivariate prediction. We research our chosen methods reliability and how to use it. More precisely, how to predict new feature values in unknown value ranges. We also try to handle data variation because of the stochastic nature of data. Meaning, how to assimilate non-functional properties.

This thesis studies data from Nokia and its multivariate models performance and predictability. Legacy data is used to build a multivariate model and its validity is measured. If model parameters are adequate, new input data is either measured or created according to the model parameters. For example, we want to increase some output values so we change the input values according to the model parameters. With the new input, the model is used to predict the corresponding output values. The iteration continues until acceptable results are accomplished or other problem is encountered. Other problems are a lack of explaining variables, possible physical limits and many more. The model is built using Projection to Latent Structures Regression or more famously Partial Least Squares (PLS).

Programming language R is used in the implementation of analysis

method. Thesis provides validation to prediction and means to see the inner relations. Inner relations are then used to optimise process/product to direction needed/wanted.

Structure of this thesis is the following: Chapter 2 has the concepts behind this thesis. Chapter 3 is about why we chose multivariate method and what method specifically. In chapter 4, we go through our analysis method in detail using mathematical theory. Chapter 5 is about experimentation and implementation. Chapter 6 is reserved for discussion of results and the future possibilities. In Chapter 7, we round up our research and conclusions.

2 Product simulation

Digitalization and Internet of Things have enabled us to perform efficiently in the digital world. This has brought up testing and performing most if not all products with simulations using machine learning and other powerful algorithms. Nowadays computers have the power and scalability making these kind of algorithms possible to successfully compute in a reasonable time. This on the other hand has raised a possibility to simulate a product virtually, without the need to actually build the product itself. And even if a product is built it can be monitored via simulated model if there is enough knowledge or data. For example, maintenance can be optimized and planned with a simulated model which "knows" when the product is on the brink of malfunction. Downtime of the system would diminish and the planned maintenance could be as efficient as possible. Data driven solutions have become viable.

2.1 Digital twin

By definition, a Digital Twin is a virtual model of a process, product or service or in other words physical asset. This pairing of the virtual and physical worlds allows analysis of data and monitoring of systems to head off problems before they even occur, prevent downtime, develop new opportunities and even plan for the future by using simulations.

The concept of Digital Twin was first introduced in 2002 at the University of Michigan by Michael Grieves for Product Lifecycle Management (PLM). Back then it was just a conceptual idea, but it already had all the elements of the Digital Twin: the real and the virtual space, data flow from the real space to the virtual space and information flow from the virtual space to the real space and also possible virtual sub-spaces. Even though the virtual space mirrors the real space, as it holds all the data which the real space has, they are meant to work together through links. [6]

At first the conceptual idea of Digital Twin was referred to as the Mirrored Spaces Model. Name changed as years passed by until 2011 when the concept was referred to as Digital Twin. This name became the norm because of its descriptiveness.

One of the earliest users of Digital Twin, NASA, saw the need of testing

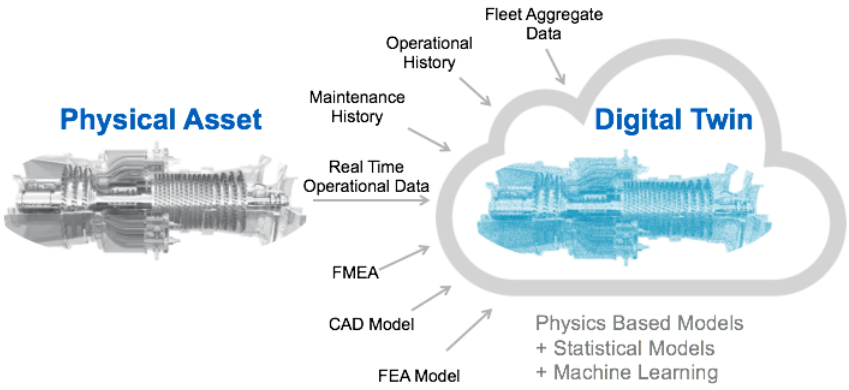

Figure 2.1: Digital Twin model [19].

and developing their systems virtually, as they could not do that physically. Digital Twin has more applications than that and now that technology and other factors have matured enough, industry has taken notice of the Digital Twin models usefulness. [12]

Digital Twin has many different manifestations such as Digital Twin Prototype (DTP), Digital Twin Instance (DTI), Digital Twin Aggregate (DTA) and more. Like name suggests, DTP describes the prototypical physical asset. It contains the informational sets necessary to describe and produce a physical version that duplicates virtual version. DTP can contain more information than what is necessary. DTI is a specific to certain product or process and is linked to that certain product through its lifetime. DTA is the aggregation of all the DTIs. As such, it may not be an independent data structure. DTA could actually be considered as a Big Data format which combines all DTI information. Digital Twin Environment (DTE) is an integrated, multi-domain physics application space for operating on Digital Twins for a variety of purposes. Two of these purposes are predictive and interrogative. The predictive purpose would be predicting future behaviour and performance of the physical product. The interrogative purpose is to be used to inquire products current or past history. For example, this way Digital Twin would know space shuttles past locations and current fuel amount without shuttles sensors. This could lessen shuttles number of components and make its composition less complex. To achieve this, the Digital Twin model of a space shuttle must

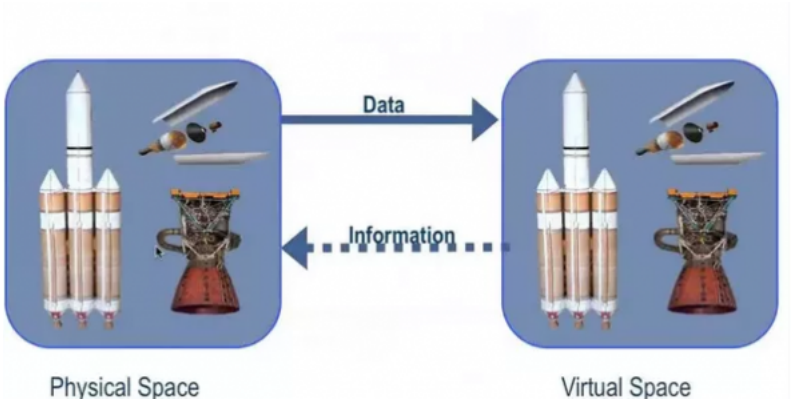

Figure 2.2: Relations in Digital Twin model [19].

be identical to the physical shuttle. Digital Twin surely has more manifestations than what are mentioned here, but these are the most common concepts to this topic. This thesis focuses on the predictive purposes of Digital Twin. [6]

As discussed above, Digital Twin can be considered as a bridge between the physical and the virtual world. Smart components that use sensors to gather data about real-time status, working condition, or position are integrated with a physical item. The components are connected to a system that receives and processes all the data the sensors monitor. This input is analysed against business and other contextual data. Results of analysis are then brought back to the physical model as information. The physical model is then modified according to the virtual models information. The virtual model becomes embedded to the physical model through sensors and this cycle of exchanging information continues.

Optimal usage of Digital Twin would be construction of virtual item, testing and improving of it, before the physical item is built. This way the physical item could be developed and optimised to the fullest before production.

Digital Twins uses are many, but it could help project management and such in an efficient way. Usually, projects mission is to produce a product that responds to customers needs and ideas. The product is designed and then created as a prototype. This prototype is then tested and validated. After many rounds of improvement, product is delivered to the customer who does some more tests. Along this road there are numerous "potholes" and

every one of them costs money and time. With Digital Twin these "potholes" could be evaded/dodged, but this is explained in detail in the next chapter.

2.2 Project development

Project is a combination of plans and people to achieve a certain aim. In business world, project is about making a product/process. Whole project planning and handling follow the Agile process, which is an iterative process. When the first iteration has been completed, new iteration starts with formers problems and new goals. This kind of process fits perfectly with simulation and Digital Twin as you will see later in this chapter.

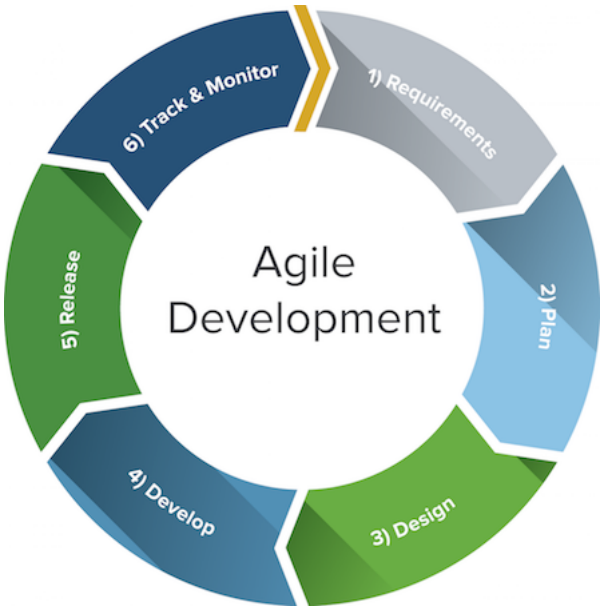

Figure 2.3: Process of Agile by [26].

Firstly, customers needs are investigated or received. Those needs become the milestones to the project. The project continues to a planning phase, where approaches and details are discussed to achieve the milestones and more. Then those plans are realised through design and development. Afterwards the product is released and monitored. Monitoring might reveal new problems but overall it is part of the cycle to gather data for other/further plans. This cycle continues spiralling until defined goals are

met. In Figure 2.3 we do not see the whole idea very clearly, but combined with Figure 2.4 we start to see how simulation and analysis fit to the project design rather well.

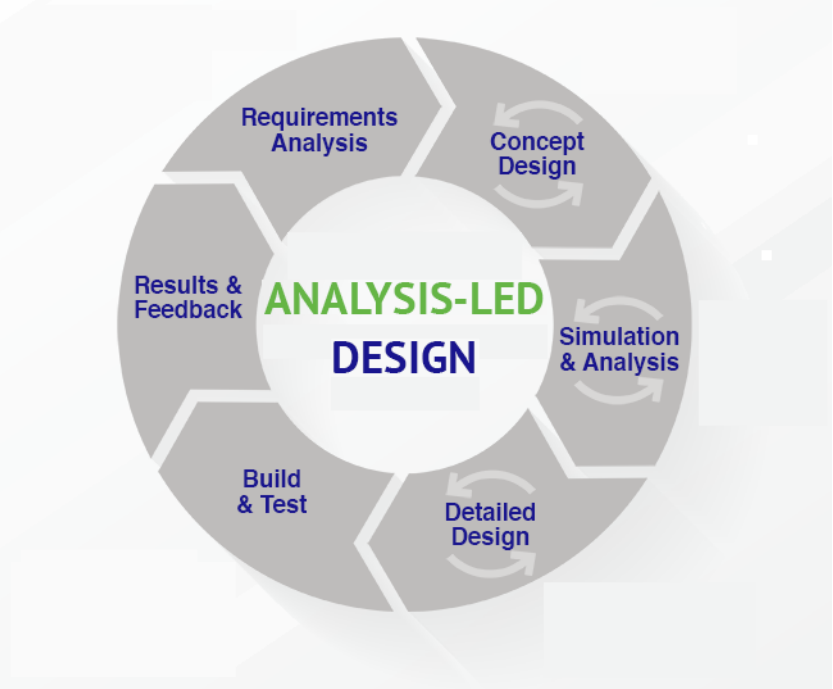

Figure 2.4: Analysis-Led Design.

Analysis-Led Design (ALD) has the same structure as Agile, but it has more practical approach. ALD begins with a requirement analysis and after it follows the iteration of the concept design. Best tools for a certain problem are revised here. Third phase uses the concept design -loops results and builds an optimal model with simulation and analysis. Model building is also an iterative loop, so that the most competent model is found. Then more detailed design is iterated. The product is then built based on all the previous results and tested. Test results and feedback decide if the cycle continues. Agile's design part includes the right half of Figure 2.4, but ALD emphasizes more on the analysis. As Figure 2.4 shows, design and development are actually more complex tasks than Figure 2.3 lets us to understand. Coming up with designs and performing simulation and analysis are iterative operations.

ALD concept is adequate as it is, but it is too lacking to fully utilize the Digital Twin. Hence, we introduce next concept that does that. In Figure 2.5 we see the same life cycle of product design as in the others with small but important changes.

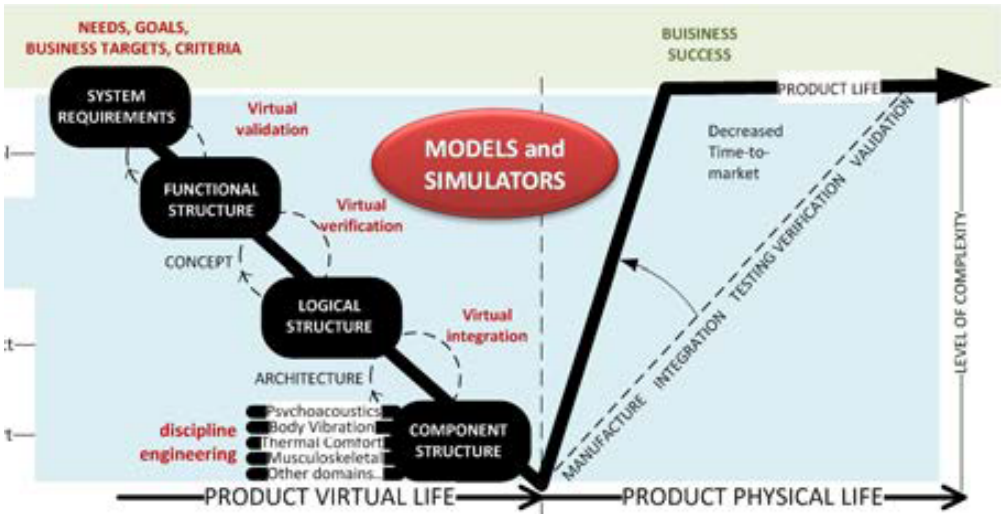

Figure 2.5: Model -based process in business processes [8].

The distinguished difference to the previous concepts here is the back and forth movement between phases which are done during products virtual life. Other design models go full round before coming back to the requirement analysis or others. Back and forth movement opens up efficient communication between departments and customers. Defined requirements can be confirmed and then redefined if needed. It is also possible for the customer to want something more now that it has been proven with models and such that the company is capable of delivering a product that fulfils the previous requirements. Possibility of not being able to fulfil requirements is also an option, but this could be noticed before producing any physical products. Back and forth iterations improve the virtual product to the fullest and then the physical product is produced based on the virtual product. Since the virtual product has been optimized, physical product should pass those tests as well. This means that the production of the physical product is much faster and there should be little to no errors. The whole process might take longer than the previous concepts, but troubleshooting becomes easier and cheaper. Most potential problems are

noticed in simulations and fixed accordingly be it insufficient part or system error. Of course, producing a product solely in the virtual space requires advanced methods of simulation and modelling.

The benefit of doing almost all development in the virtual space is that producing the physical product is rather expensive. If this is done after every little upgrade, costs would be significant. What makes it worse is the fact that not every prototype can be reused in the next revision, making it nothing but trash. Upgrading the virtual product on the other hand is more manageable. Improving code or algorithm does not create as much waste. Testing and observing the product can also be dangerous or impossible in physical environment. Like in NASA's case testing the prototype of a space shuttle in space is quite expensive, but also difficult to some extend, since sensors and cameras would have to be placed in every spot to see what happened.

The virtual life of a product can be thought as a construction of a Digital Twin. Data and models are used to imitate the physical product. Data can be from previous revisions or other similar sources if deemed relevant. Even though, Digital Twins definition dictates that Digital Twin is a virtual model of a physical asset, it is not necessary to have that physical asset. With enough knowledge and data about physical asset, Digital Twin can be made.

2.3 Continuous development

MegaM@Rt^2 -projects (https://megamart2-ecsel.eu/) goal is to produce a framework incorporating methods and tools for continuous development and validation leveraging the advantages in scalable model-based methods. This framework consists of parts seen in Figure 2.6. Simulation and Digital Twin terms fit the description well and thus, are used to reach projects goal.

Frameworks continuous development is maintained by a system, data gathering and validation like in Figure 2.6. System is made of multiple models which analyse data. In our case the system would predict behaviour with new input values using past data. Prediction results are expected behaviour of the real asset. New input data is then used in real machinery, collecting output data. Output data goes to the system to validate models success rate. Depending on the results, models are modified and the last output data is added to the past data increasing the legacy data with every run.

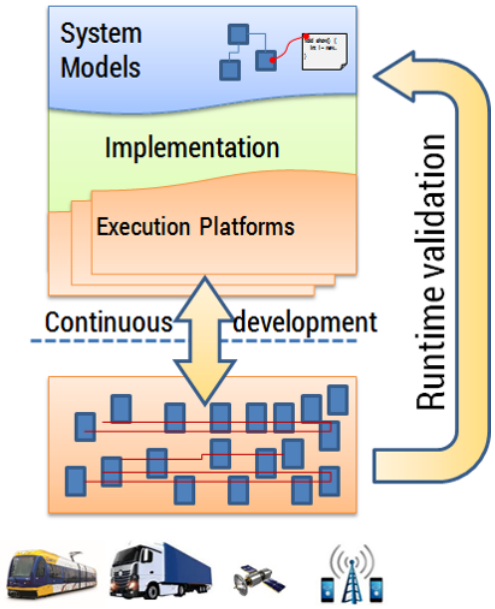

Figure 2.6: MegaM@Rt^2 runtime approach [13].

Everything here has one thing in common; data, which acts as the base and fuel. Without data, Digital Twin is impossible to build. Without data, simulations can not be improved. Since this is the case, we must consider data-driven solutions. Human intuition can only take us so far, while data-driven models have concrete evidence on the data behaviour. Thus, mathematical models can provide us with a solution. More specifically, multivariate analysis seems to be the correct path.

3 Multivariate analysis

Multivariate analysis (MVA) is analysis of more than a few variables. Unlike univariate analysis which focuses on one variable, MVA analyses multiple variables together. Depending on a method, MVA can use correlation or such to study variables and their relations to each other. There are many methods to perform MVA depending on the data at hand. [21]

Analysing only one variable can be efficient and fast, but that one variable may not tell enough. For example, think of human senses. By using all our senses, we are in fact performing multivariate analysis. We see, feel and so on to notice various things. If we tried to observe our surroundings with one sense, like univariate analysis does, our understanding would be very limited. And this is true with science, economics and others. One cannot predict weather with seasons alone.

Often variables are not independent of each other. This relation is sometimes rather hard to notice with univariate methods. When it is cloudy then sometimes it rains but without clouds it hardly ever rains. Weather being cloudy and when it rains have some correlation between them and other variables. These kind of relations are actually important when inspecting complex systems. Many variables contribute to a certain behaviour. System error could be caused by two variables that both reached a certain limit. Individually these two variables could have bigger range, but simultaneously they break the system.

Using multivariate analysis method over univariate seems appropriate, since we are dealing with complex systems. Also interpreting variable relations adds value to multivariate analysis over univariate.

3.1 Multivariate prediction

Multivariate prediction is usually done with multivariate methods, which form a regression model between independent (predictor) and dependent (predicted) variables.

Multivariate prediction has many algorithms that are used in different situations like machine learning methods, multivariate linear regression, etc. Traditional methods are quite accurate, even with limited amount of data, but they are often so called "black box" methods or they overfit to a

training data too easily. With "black box" we mean that it is not known what happens and for what reason inside a chosen method. There are situations, where it is imperative to know correlations and explaining factors. Especially in simulation, where explaining factors could reduce number of test and potentially optimize the process. Overfitting, on the other hand, means that the built model fits to the training data too well impairing models predictive power.

We chose to use PLS as our method of analysis because of its success in chemistry and its fundamental qualities. Multivariate data tends to have more variables than observations and those variables usually correlate with each other i.e. multicollinearity. In these cases other methods perform poorly or not at all if multicollinearity is too high. Otherwise as the observations increase, the accuracy of PLS improves. Other problem is models overfitting to data it is built from. PLS is observed to not overfit so easily than similar methods [11] [22]. Overfitting is related to models prediction rate with a training data. Data model is built from the training data. If the model overfits, the model might lose its generality. Other data, that we use in testing, validation or prediction, might not share the same relations as the training data.

Other similar methods are Redundancy Analysis (RDA) and Principal Component Regression (PCR). RDA was developed by Van den Wollenberg (1977) and focuses more on the output variation than input. Lack of inputs influence when defining variable components erodes prediction accuracy. [20] [29]

PCR is a regression analysis technique that is based on the Principal Component Analysis (PCA). As such, PCR chooses the variable components according to the maximum variation in the input data. This can yield important information about the factor space, but that information might not be associated with correct shape of prediction. [10]

PLS is somewhat between these two as the components are calculated with both input and output variation in mind. PLS is considered to be better at predictive analysis than the two other candidates [18]. Another way to relate the three techniques is to note that PCR is based on the spectral decomposition of X^TX, where X is the matrix of factor (input) values; RDA is based on the spectral decomposition of $\hat{Y^T}\hat{Y}$, where $\hat{Y}$ is the matrix of (predicted) response values; and PLS is based on the singular value decomposition of X^TY. [20]

3.2 Proposed solution/method

We are to predict system behaviour with the Digital Twin concept. To make that happen, we have the following proposal. Use multivariate prediction method to perform prediction with data-driven models. Our chosen method PLS makes a model out of data, which can then be used to predict new output values using new input values. In this sense PLS works like a Digital Twin. Data from a product or system is fed to the PLS algorithm that forms a model (or a Digital Twin) based on the incoming data. Then the new predicted output values are brought back to the physical product as information. In short, PLS creates an environment for the Predictive Twin. If the validation parameters of the model are good enough, that information can be used. Then we know how the product or system behaves with new input values. Even if the estimates are not exact, they can be used as a hint.

PLS has an option to optimise model variables. Depending on how the input and output variables correlate with each other, it is possible to see which input variables must be modified to change a certain output variables. Other way around does not matter because the output variables are dependant of the input variables and the input variables are independent variables.

Here we must remember that the capability of the model is much dependant on the data used to built it. Too little data might not capture overall behaviour of a system. Data can also be inadequate, meaning there are not enough measured variables or no meaningful variables at all. In latter case other analysis method should be considered. If there are not enough measured variables, then the data collection might need some re-evaluation.

PLS can be used in many fields because the only crucial part is data. This also makes it easier to update a model by rebuilding it with new/more data.

3.3 Data

Data is a collection of information. Data can be from multiple sources and of multiple observations. Data can be divided into a numeric and non-numeric data. Numeric data is all the data with only numbers. Non-numeric data is the rest of the data like strings of letters and/or numbers. Non-numeric data

is hard to analyse mathematically since we cannot use arithmetic operations on it.

Because, we are considering data-driven models, the quality of data is an important factor in analysis.

3.3.1 Data quality

Data quality refers to the overall utility of a dataset(s). Data is considered high quality if it fits to intended purpose or if it represents the real-world well. There are many definitions to data quality. [5]

Since, our method is data-driven, we must make sure that the used data is of adequate quality. If the data is garbage, then the results will also be garbage like in Figure 3.1. To ensure that data quality is top-notch, the

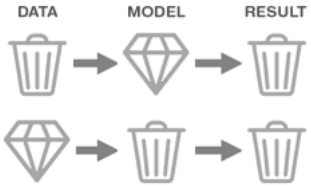

Figure 3.1: Relations between data and model quality [27].

process of data gathering should be well planned and supervised. The purpose to which the data is gathered needs to be well defined. Data gathering is then planned to match defined purposes. This way most if not all required data will be gathered. Out of context data can be combined to the main data if deemed necessary.

All sorts of problems can be encountered when gathering data. There can be problems with faulty sensors, multiple data formats and so on. These problems can create biased data, making it useless and if analysed can produce wrongful results. Faulty results on the other hand lead to wrong decisions which complicate things even further.

There is also a problem with the relevancy of data. As we use legacy data, the system now and the system before might be too different so that the data between those systems is no longer compatible. This is a serious concern to the model built with both these datasets. Old data affects the model and models bias might increases. This problem should be evaluated with an expert's knowledge.

4 Mathematical theory

This chapter roughly follows the book "Kernel Methods for Pattern Analysis" [9] from the relative parts. The book in question contains more on the topic and is a great source for further studies. Other articles and books back up the information [1] [10] [11] [22] [28] [16].

Partial Least Squares (PLS) is a widely used regression modelling technique which models the relationship between two datasets. The input dataset is denoted as X and the output as Y. PLS finds the directions in the predictor variables, X, and the responses, Y, corresponding to the maximum covariance between X and Y. PLS model parameters are used for analysis of data and for regression. With regression, we can predict data values other than the observed data. We start with some notations which are necessary for our theory section and continue through PCA to PLS model validation. This is done so that the reader gets a firm grasp on the theory behind PLS modelling.

4.1 Pre-process and notations

Matrix X is $l \times n$ consisting of column vectors $X = [\boldsymbol{x}_1 \ \dots \ \boldsymbol{x}_n]$ where each vector is $\boldsymbol{x}_i = (x_1, \dots, x_l)^T, \ i = 1, 2, \dots, n$. The transpose of matrix X is denoted as X^T. The number of columns n is the amount of features and l is the number of observations in the matrix. As a standard procedure, we perform normalisation in which each column of matrix X is centred by the columns mean and divided by the columns standard deviation

$$\frac{\boldsymbol{x}_i - \mu_i}{\sigma_i}, \ i = 1, 2, \dots, n,$$

where μ_i and σ_i are mean and standard deviation values of vector $\boldsymbol{x}_i$. Note that μ_i and σ_i are scalars. This is done in order that no variable outweighs the other because of its value scale. We standardize the values of different scales to a notional common scale. For example, the scale of weight is somewhat different than the scale of height. This is fine because these scales are similar but if the scale difference is too big it will lean our analysis towards variables with a larger scale. This is not necessarily a bad thing as long as it is noticed.

There are cases where standardisation is not needed if variables are in close enough scales. In these cases, mean-centring is the only pre-processing

needed. Mean-centring matrix X follows a process

$$\boldsymbol{x}_i - \mu_i, \ i = 1, 2, \ldots, n.$$

This is a minimum condition to our PLS analysis.

Since, regular PLS falls into a linear regression category, if there are non-linear data relations, then a PLS model might not perform all that well. But if these non-linearities are noticed, data can be transformed to be linear. There are many methods like exponential, logarithmic, etc. data transforms. For example, using exponential transform, we take the exponent of all variable values and fit a linear model to the transformed data. If the validation measures of the model improve, then the data has become more linear than before. Without prior knowledge, these methods are hard to use, especially in multivariate cases, because of the amount of variables. These are usually executed via trial and error to see whether the model parameters get better. Combinations in trial and error method are many and might not make the regression model significantly better. Before the data are transformed, residual plots can provide a hint to the need of data transform. [11]

4.1.1 Feature space

In this section we show dataset properties in feature space and introduce projection of data onto a subspace. Feature space is a space where our variables are located. Basic knowledge is needed to understand and form upcoming theory and proofs. For some column vector $\boldsymbol{x}$ in matrix X, the L^2 norm of a vector is given by

$$\begin{aligned} \|\boldsymbol{x}\|_2 &= \sqrt{\|\boldsymbol{x}\|^2} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \\ &= \sqrt{\boldsymbol{x}^T \boldsymbol{x}} = \sqrt{x_1^2 + \ldots + x_l^2}. \end{aligned}$$

The L^2 norm measures the length of a vector. To normalise a feature vector, we divide the vector by its length

$$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}.$$

The normalised vector $\hat{\boldsymbol{x}}$ is of length 1 and is also called a unit vector.

The projection $P_{\boldsymbol{w}}(\boldsymbol{x})$ of a vector $\boldsymbol{x}$ onto some vector $\boldsymbol{w}$ is given as

$$P_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{\langle \boldsymbol{w}, \boldsymbol{x} \rangle}{\|\boldsymbol{w}\|^2} \boldsymbol{w}.$$

With this we can compute projections of vectors in the feature space. Projection P_V, where $V = [\boldsymbol{w_1} \ \ldots \ \boldsymbol{w_k}]$ is the basis of the projection, satisfies

$$P_V(\boldsymbol{x}) = P_V^2(\boldsymbol{x}) \quad \text{and} \quad \big\langle P_V(\boldsymbol{x}), \boldsymbol{x} - P_V(\boldsymbol{x}) \big\rangle = 0,$$

with its dimension dim (P_V) given by the dimension of the image of P_V. The orthogonal projection to P_V is given by

$$P_V^{\perp}(\boldsymbol{x}) = \boldsymbol{x} - P_V(\boldsymbol{x}),$$

which projects the data onto the orthogonal complement of the image of P_V, so that dim (P_V) + dim$(P_V^{\perp}) = n$, the dimension of the feature space. The orthogonal projection $P_V^{\perp}$ satisfies the rules above. The projection $P_{\boldsymbol{w}}(\boldsymbol{x})$ of a vector $\boldsymbol{x}$ onto some vector $\boldsymbol{w}$ can be expressed differently if we assume that $\boldsymbol{w}$ is normalised

$$P_{\boldsymbol{w}}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle \boldsymbol{w} = \boldsymbol{w}^T \boldsymbol{x} \boldsymbol{w} = \boldsymbol{w} \boldsymbol{w}^T \boldsymbol{x}.$$

Rearranging the equation changes $\boldsymbol{w}\boldsymbol{w}^T$ to become a matrix. The orthogonal projection can be formed similarly,

$$P_{\boldsymbol{w}}^{\perp}(\boldsymbol{x}) = (I - \boldsymbol{w}\boldsymbol{w}^T)\boldsymbol{x}.$$

If $\boldsymbol{w}$ is not normalised then the projection would become

$$P_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{\langle \boldsymbol{w}, \boldsymbol{x} \rangle}{\langle \boldsymbol{w}, \boldsymbol{w} \rangle} \boldsymbol{w} = \frac{\boldsymbol{w}^T \boldsymbol{x}}{\boldsymbol{w}^T \boldsymbol{w}} \boldsymbol{w}.$$

The deflation of matrix $X^T X$, meaning the removal of some eigenvalue λ of the matrix $X^T X$ using the corresponding eigenvector $\boldsymbol{w}$, is the same as projecting the data using $P_{\boldsymbol{w}}^{\perp}$. In this way we get a new data matrix $\tilde{X}$.

$$\tilde{X} = X(I - \boldsymbol{w}\boldsymbol{w}^T)^T = X(I - \boldsymbol{w}\boldsymbol{w}^T). \tag{4.1}$$

This combined with the fact that $X^T X \boldsymbol{w} = \lambda \boldsymbol{w}$ gets us

$$\begin{aligned}\tilde{X}^T \tilde{X} &= (I - \boldsymbol{w}\boldsymbol{w}^T) X^T X (I - \boldsymbol{w}\boldsymbol{w}^T) \\ &= X^T X - \boldsymbol{w}\boldsymbol{w}^T X^T X - \boldsymbol{w}\boldsymbol{w}^T X^T X + \boldsymbol{w}\boldsymbol{w}^T X^T X \boldsymbol{w}\boldsymbol{w}^T \\ &= X^T X - \lambda \boldsymbol{w}\boldsymbol{w}^T - \lambda \boldsymbol{w}\boldsymbol{w}^T + \lambda \boldsymbol{w}\boldsymbol{w}^T \boldsymbol{w}\boldsymbol{w}^T \\ &= X^T X - \lambda \boldsymbol{w}\boldsymbol{w}^T,\end{aligned}$$

where λ is an eigenvalue corresponding to a normalised vector $\boldsymbol{w}$.

We now consider how to find an orthonormal basis for such a subspace. More generally we seek a subspace that fits the data in the sense that the distances between data items and their projections into the subspace are small.

Well-known method to do this is the Gram-Schmidt orthonormalisation. Given a sequence of linearly independent vectors the method creates the basis by orthogonalising each vector to all of the earlier vectors. Hence, if we are given the vectors

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l,$$

the first basis vector is

$$\boldsymbol{q}_1 = \frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|}.$$

The next vectors are then obtained by subtracting the projection of $\boldsymbol{x}_i$ to previous basis vectors from $\boldsymbol{x}_i$ to ensure orthogonality.

$$\boldsymbol{x}_i \to \boldsymbol{x}_i - \sum_{j=1}^{i-1} \langle \boldsymbol{q}_j, \boldsymbol{x}_i \rangle \boldsymbol{q}_j = (I - Q_{i-1} Q_{i-1}^T) \boldsymbol{x}_i,$$

where Q_i is the matrix whose columns are the first i vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_i$. The matrix $(I - Q_i Q_i^T)$ is a projection matrix onto the orthogonal complement of the space spanned by the first i vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_i$. Taking norm of our projection and presenting it as follows

$$v_i = \|(I - Q_{i-1} Q_{i-1}^T) \boldsymbol{x}_i\|,$$

we can express basis vectors

$$\boldsymbol{q}_i = v_i^{-1} (I - Q_{i-1} Q_{i-1}^T) \boldsymbol{x}_i.$$

With these $\boldsymbol{x}_i$ gets following form

$$\begin{aligned}\boldsymbol{x}_i &= Q_{i-1}Q_{i-1}^T\boldsymbol{x}_i + v_i\boldsymbol{q}_i = Q_i \begin{pmatrix} Q_{i-1}^T\boldsymbol{x}_i \\ v_i \end{pmatrix} \\ &= Q \begin{pmatrix} Q_{i-1}^T\boldsymbol{x}_i \\ v_i \\ \mathbf{0}_{l-i} \end{pmatrix} = Q\boldsymbol{r}_i,\end{aligned}$$

where matrix $Q = Q_l$ contains vectors $\boldsymbol{q}_i$ as columns. By expanding to matrix X^T expression becomes

$$X^T = QR,$$

suggesting to use QR-decomposition to solve the basis vectors, since matrices Q and R are the same in both cases. In "Partial least squares" section we use Gram-Schmidt orthonormalisation as a frame to compute the basis vectors to the new feature space.

4.2 Principal components analysis

Principal component analysis focuses on directions that have the maximum variance in feature space. Data has been centred as a pre-process mentioned in the beginning of this chapter. To find the directions of maximum variance we must first solve projections variance. The variance of the projection onto a vector $\boldsymbol{w}$ is calculated using expected mean of the squared deviation from the mean. The expected mean value of the projection onto a vector $\boldsymbol{w}$ is

$$\hat{E}\left[P_{\boldsymbol{w}}(\boldsymbol{x})\right] = \frac{1}{l}\sum_{i=1}^{l} P_{\boldsymbol{w}}(\boldsymbol{x}_i).$$

The variance of the projection onto a vector $\boldsymbol{w}$ is then

$$\sigma_{\boldsymbol{w}}^2 = \hat{E}\left[(\|P_{\boldsymbol{w}}(\boldsymbol{x})\| - \mu_{\boldsymbol{w}})^2\right] = \hat{E}\left[\|P_{\boldsymbol{w}}(\boldsymbol{x})\|^2\right] = \frac{1}{l}\sum_{i=1}^{l} \|P_{\boldsymbol{w}}(\boldsymbol{x}_i)\|^2, \qquad (4.2)$$

where $\|P_{\boldsymbol{w}}(\boldsymbol{x})\| = \boldsymbol{w}^T\boldsymbol{x}/(\boldsymbol{w}^T\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x}$ is the norm of the projection of points onto a space spanned by $\boldsymbol{w}$, when $\boldsymbol{w}$ is normalized. Since our data

is centred, the mean value of data is $\mu_w = 0$. Equation 4.2 can be followed further

$$\frac{1}{l}\sum_{i=1}^{l} \|P_{\boldsymbol{w}}(\boldsymbol{x}_i)\|^2 = \hat{E}[\boldsymbol{w}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{w}] = \boldsymbol{w}^T\hat{E}[\boldsymbol{x}\boldsymbol{x}^T]\boldsymbol{w}$$
$$= \frac{1}{l}\boldsymbol{w}^T X^T X\boldsymbol{w} = \boldsymbol{w}^T C_{xx}\boldsymbol{w},$$

where $C_{xx} = \frac{1}{l}X^T X$ is the covariance matrix of the data sample. With eigenvalue decomposition we get the relation $lC_{xx} = X^T X = U\Lambda_n U^T$. To find the direction that maximises variance, next problem must be solved

$$\begin{aligned} &\max_{\boldsymbol{w}} && \boldsymbol{w}^T C_{xx}\boldsymbol{w} \\ &\text{subject to} && \|\boldsymbol{w}\|_2 = 1. \end{aligned} \tag{4.3}$$

Eigenvectors for maximising variance follow from quotient

$$\rho(\boldsymbol{w}) = \frac{\boldsymbol{w}^T C_{xx}\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{w}}.$$

The reason for using eigenvectors to define the directions of maximum variance is that the eigenvalue corresponds to the variance of data. By finding the largest eigenvalue and eigenvector respect to that eigenvalue, we have then found the direction of maximum variance in the data. The solution of 4.3 is the direction that maximises $\rho(\boldsymbol{w})$. This follows from the *Rayleigh quotient*

$$\frac{\boldsymbol{w}^T C_{xx}\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{w}} = \lambda\frac{\boldsymbol{w}^T\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{w}} = \lambda. \tag{4.4}$$

We can conclude that $\rho(\boldsymbol{w}) = \lambda$ and that $\boldsymbol{w}$ is an eigenvector corresponding to the largest eigenvalue of λ. *Rayleigh quotient* is subjected to Courant-Fisher theorem 1 so that we find the largest eigenvalues with their corresponding eigenvectors to equation 4.3.

Theorem 1 (Courant-Fisher)**.** *If $A \in \mathbb{R}^{n\times n}$ is symmetric with eigenvalues $\lambda_1 \leq \ldots \leq \lambda_n$, then for $k = 1, \ldots, n$, the kth eigenvalue $\lambda_k(A)$ of the matrix A satisfies*

$$\lambda_k(A) = \max_{dim(T)=k}\min_{\boldsymbol{v}\neq 0, \boldsymbol{v}\in T}\frac{\boldsymbol{v}^T A\boldsymbol{v}}{\boldsymbol{v}^T\boldsymbol{v}} = \min_{dim(T)=n-k+1}\max_{\boldsymbol{v}\neq 0, \boldsymbol{v}\in T}\frac{\boldsymbol{v}^T A\boldsymbol{v}}{\boldsymbol{v}^T\boldsymbol{v}},$$

with extrema achieved by corresponding eigenvector.

The direction that has second largest variance in the orthogonal subspace can be found, by looking for the largest eigenvector in the matrix obtained by deflating the matrix C_{xx} with respect to $\boldsymbol{w}$. This gives the eigenvector of C_{xx} corresponding to the second largest eigenvalue. Repeating this step shows that the mutually orthogonal directions of maximum variance in order of decreasing size are given by the eigenvectors of C_{xx}.

In fact, the size of the eigenvalue is equal to the variance in the chosen direction. This way explained variance can be expressed as a percentage value by dividing our eigenvalue with the sum of all eigenvalues.

Since eigenvectors are unaffected by rescaling and it only rescales eigenvalues, matrix $lC_{xx} = X^T X$ can be analysed so that we find the directions of maximum variance ie. eigenvectors. By analysing the matrix $X^T X$ we can use our knowledge of projections. The first eigenvalue of the matrix lC_{xx} equals to the sum of the squares of the projections of the data into the first eigenvector in the feature space. From the Courant-Fisher theorem 1 follows

$$\begin{aligned}\lambda_1(lC_{xx}) = \lambda_1(X^T X) &= \max_{dim(T)=1} \min_{\boldsymbol{u}\neq 0, \boldsymbol{u}\in T} \frac{\boldsymbol{u}^T X^T X \boldsymbol{u}}{\boldsymbol{u}^T \boldsymbol{u}} \\ &= \max_{\boldsymbol{u}\neq 0} \frac{\boldsymbol{u}^T X^T X \boldsymbol{u}}{\boldsymbol{u}^T \boldsymbol{u}} = \max_{\boldsymbol{u}\neq 0} \frac{\|X\boldsymbol{u}\|^2}{\|\boldsymbol{u}\|^2} = \max_{\boldsymbol{u}\neq 0} \sum_{i=1}^{l} P_{\boldsymbol{u}}(\boldsymbol{x}_i)^2 \\ &= \sum_{i=1}^{l} \|\boldsymbol{x}_i\|^2 - \min_{\boldsymbol{u}\neq 0} \sum_{i=1}^{l} \|P_{\boldsymbol{u}}^{\perp}(\boldsymbol{x}_i)\|^2, \end{aligned} \tag{4.5}$$

where $P_{\boldsymbol{u}}^{\perp}$ is the projection of $\boldsymbol{x}$ into the space orthogonal to $\boldsymbol{u}$. The last step is possible because of the Pythagoras's theorem and the fact that projections are orthogonal,

$$\|P_{\boldsymbol{u}}(\boldsymbol{x}_i)\|^2 = \|\boldsymbol{x}_i\|^2 - \|P_{\boldsymbol{u}}^{\perp}(\boldsymbol{x}_i)\|^2.$$

Do note that vectors $\boldsymbol{u}_i$ are columns of matrix U of the eigenvalue decomposition

$$X^T X = U \Lambda U^T.$$

The ith case finds the ith eigenvalue. This follows similarly to Courant-

Fisher theorem 1

$$\lambda_i(lC_{xx}) = \lambda_i(X^TX) = \max_{dim(T)=i} \min_{\boldsymbol{u}\neq 0, \boldsymbol{u}\in T} \frac{\boldsymbol{u}^T X^T X \boldsymbol{u}}{\boldsymbol{u}^T \boldsymbol{u}}$$
$$= \max_{dim(T)=i} \min_{\boldsymbol{u}\neq 0, \boldsymbol{u}\in T} \sum_{j=1}^{l} P_{\boldsymbol{u}}(\boldsymbol{x}_j)^2 = \sum_{j=1}^{l} P_{\boldsymbol{u}_i}(\boldsymbol{x}_j)^2,$$

that is, the sum of the squares of the projections of the data in the direction of the ith eigenvector $\boldsymbol{u}_i$ in the feature space. Now if we want to project into the space U_k defined by the first k eigenvectors, we get

$$\sum_{i=1}^{k} \lambda_i = \sum_{i=1}^{k} \sum_{j=1}^{l} P_{\boldsymbol{u}_i}(\boldsymbol{x}_j)^2 = \sum_{j=1}^{l} \sum_{i=1}^{k} P_{\boldsymbol{u}_i}(\boldsymbol{x}_j)^2 = \sum_{j=1}^{l} \|P_{U_k}(\boldsymbol{x}_j)\|^2, \qquad (4.6)$$

where $P_{U_k}(\boldsymbol{x})$ denotes the orthogonal projection of $\boldsymbol{x}$ into the subspace $U_k = \{\boldsymbol{u_1} \dots \boldsymbol{u}_k\}$. If $k = N$, then the projection becomes the identity

$$\sum_{i=1}^{N} \lambda_i = \sum_{j=1}^{l} \|P_{U_N}(\boldsymbol{x}_j)\|^2 = \sum_{j=1}^{l} \|\boldsymbol{x}_j\|^2.$$

The k first principal components minimise the difference between the projection and the original data

$$\sum_{j=1}^{l} \|P_{U_k}(\boldsymbol{x}_j)\|^2 = \sum_{j=1}^{l} \|\boldsymbol{x}_j\|^2 - \sum_{j=1}^{l} \|P_{U_k}^{\perp}(\boldsymbol{x}_j)\|^2,$$

where $dim\ U = k$. Proof to this can be found from [9] page 148.

As such, PCA forms the principal axes of the training data and projects the data into the space defined by a set of eigenvectors. New coordinates are known as the principal coordinates with eigenvectors referred to as the principle axes. The advantage of PCA is that as it uses eigenvalues and eigenvectors, explained variance of each component can be measured. This new space can be used for regression and other analyses.

4.3 Principal component regression

Standard multivariate linear regression becomes impossible to solve if there is high multicollinearity in the input matrix. Multicollinearity means that

two or more variables are highly correlated. Because of the multicollinearity, coefficient matrix B has no explicit solution when using the ordinary least-squares estimator. But if we use the space and the features from the PCA we can overcome multicollinearity and perform least squares regression.

In our case we have multiple response variables forming $l \times m$ matrix Y, so the formula is

$$\min_B \|XB - Y\|_F^2,$$

which involves the squared Frobenius norm instead of the squared L^2 norm. The Frobenius norm is the square root of the sum of the absolute squares of the matrix elements. For matrix A with size $m \times n$ the Frobenius norm is

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

We also use SVD of $X^T = U\Sigma V^T$ during estimating the coefficient B. Matrices U size $n \times n$ and V size $l \times l$ are the singular matrices of X^T and Σ size $n \times l$ is the diagonal matrix with eigenvalues. By inverting the diagonal values of Σ we get Σ^{-1}. This is also true for Σ^T because it is a diagonal matrix.

As the name Principal Component Regression (PCR) suggests, by performing PCA we find a new space where we perform the regression. We take the first k eigenvectors of X^TX as features. These features form the principal axes of our feature space. So instead of using X, we use new scores XU_k which are in the new space defined by PCA. Matrix U_k is the first k columns of the matrix U from SVD $X^T = U\Sigma V^T$. We leave Y as it is. The least squares regression using PCR becomes the following

$$\begin{aligned}\min_B \|XU_kB - Y\|_F^2 &= \min_B \langle XU_kB - Y, XU_kB - Y\rangle_F \\ &= \min_B(\mathrm{tr}(B^TU_k^TX^TXU_kB) - 2\mathrm{tr}(B^TU_k^TX^TY) \\ &\quad + \mathrm{tr}(Y^TY)) \\ &= \min_B(\mathrm{tr}(B^TU_k^TX^TXU_kB) - 2\mathrm{tr}(B^TU_k^TX^TY)).\end{aligned}$$

The regression coefficients are given by U_kB and the minimum is found by computing the gradient with respect to B and setting to zero. This results

$$U_k^TX^TXU_kB = U_k^TX^TY. \tag{4.7}$$

By performing SVD on X^T we get the regression coefficient B as

$$B = \bar{\Sigma}_k^{-2} U_k^T X^T Y,$$

where $\bar{\Sigma}_k^{-2}$ is a square matrix containing the first k columns of Σ_k^{-2}. This way we only need to compute matrices Σ and U to solve B.

As seen above, PCR is quite simple yet effective method. Though, the problem of not taking into account features of Y will not make the most efficient linear regression coefficients regarding prediction. It is still important to note that good qualities of PCA are present in PCR, like noise reduction and dimension reduction.

4.4 Partial least squares

PLS is an extension of the Principal Component Analysis (PCA) and is in a bit different aspect than Principal Component Regression which performs least squares regression in the PCA defined space. Instead of the variance of inputs, it might be more important to observe the covariances of the inputs and the outputs. And that is precisely what PLS uses to determine the new feature space. Derived feature space is then used to perform regression.

The covariance between two datasets that lay in different spaces can be difficult to measure. That is why we project the data onto a direction $\boldsymbol{w}$. In this case we have two directions $\boldsymbol{w_x}$ and $\boldsymbol{w_y}$. These form random variables $\boldsymbol{w_x}^T\boldsymbol{x}$ and $\boldsymbol{w_y}^T\boldsymbol{y}$, which we use to asses the relation between $\boldsymbol{x}$ and $\boldsymbol{y}$.

We must first find the directions of maximum covariance between two separate spaces X and Y. Here is the covariance of our two random variables.

$$\hat{E}[\boldsymbol{w_x}^T\boldsymbol{x}\boldsymbol{w_y}^T\boldsymbol{y}] = \hat{E}[\boldsymbol{w_x}^T\boldsymbol{x}\boldsymbol{y}^T\boldsymbol{w_y}] = \boldsymbol{w_x}^T\hat{E}[\boldsymbol{x}\boldsymbol{y}^T]\boldsymbol{w_y} = \boldsymbol{w_x}^T C_{xy}\boldsymbol{w_y},$$

where C_{xy} is the sample covariance matrix between X and Y. We can also write

$$C_{xy} = \hat{E}[\boldsymbol{x}\boldsymbol{y}^T] = \frac{1}{l}\sum_{i=1}^{l} \boldsymbol{x}_i\boldsymbol{y}_i^T = \frac{1}{l}X^T Y,$$

where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are row vectors of their respective matrices. To find directions $\boldsymbol{w_x}$ and $\boldsymbol{w_y}$ which maximise the covariance, we consider the

following way,

$$\begin{aligned} &\max_{\boldsymbol{w_x},\boldsymbol{w_y}} \qquad C(\boldsymbol{w_x},\boldsymbol{w_y}) = \boldsymbol{w_x}^T C_{xy} \boldsymbol{w_y} = \boldsymbol{w_x}^T \frac{1}{l} X^T Y \boldsymbol{w_y} \\ &\text{subject to} \qquad\qquad\qquad\qquad \|\boldsymbol{w_x}\|_2 = \|\boldsymbol{w_y}\|_2 = 1. \end{aligned} \tag{4.8}$$

This problem is similar to the PCA maximum variance problem.

Proposition 1. *The directions that solve the maximal covariance optimisation are the first singular vectors $\boldsymbol{w_x} = \boldsymbol{u}_1$ and $\boldsymbol{w_y} = \boldsymbol{v}_1$ of the singular value decomposition of C_{xy}*

$$C_{xy} = U\Sigma V^T;$$

the value of the covariance is given by the corresponding singular value σ_1.

Proof. Using the singular value decomposition of C_{xy} and taking into account that U and V are orthogonal matrices so that, for example, $\|V\boldsymbol{w}\| = \|\boldsymbol{w}\|$ and any $\boldsymbol{w_x}$ can be expressed as $U\boldsymbol{u_x}$ for some $\boldsymbol{u_x}$, the solution to the problem 4.8 becomes

$$\begin{aligned} \max_{\boldsymbol{w_x},\boldsymbol{w_y}:\|\boldsymbol{w_x}\|_2=\|\boldsymbol{w_y}\|_2=1} C(\boldsymbol{w_x},\boldsymbol{w_y}) &= \max_{\boldsymbol{u_x},\boldsymbol{v_y}:\|U\boldsymbol{u_x}\|_2=\|V\boldsymbol{v_y}\|_2=1} (U\boldsymbol{u_x})^T C_{xy} V\boldsymbol{v_y} \\ &= \max_{\boldsymbol{u_x},\boldsymbol{v_y}:\|\boldsymbol{u_x}\|_2=\|\boldsymbol{v_y}\|_2=1} \boldsymbol{u_x}^T U^T U\Sigma V^T V\boldsymbol{v_y} \\ &= \max_{\boldsymbol{u_x},\boldsymbol{v_y}:\|\boldsymbol{u_x}\|_2=\|\boldsymbol{v_y}\|_2=1} \boldsymbol{u_x}^T \Sigma \boldsymbol{v_y}. \end{aligned}$$

The last line has a maximum of the singular value σ_1, when we take $\boldsymbol{u_x} = \boldsymbol{e}_1$ and $\boldsymbol{v_y} = \boldsymbol{e}_1$ the first unit vector in optimal case. Thus, the solution is to take $\boldsymbol{w_x} = \boldsymbol{u}_1 = U\boldsymbol{e}_1$ and $\boldsymbol{w_y} = \boldsymbol{v}_1 = V\boldsymbol{e}_1$, respectively. □

Following Proposition 1, we can now compute the directions that maximise the covariance. To identify more directions, deflation is used. Similarly to equation 4.1

$$X \longleftarrow X(I - \boldsymbol{u}_1\boldsymbol{u}_1^T) \quad \text{and} \quad Y \longleftarrow Y(I - \boldsymbol{v}_1\boldsymbol{v}_1^T).$$

The covariance matrix is then as follows

$$\begin{aligned} \frac{1}{l}(I - \boldsymbol{u}_1\boldsymbol{u}_1^T)X^T Y(I - \boldsymbol{v}_1\boldsymbol{v}_1^T) &= (I - \boldsymbol{u}_1\boldsymbol{u}_1^T)U\Sigma V^T(I - \boldsymbol{v}_1\boldsymbol{v}_1^T) \\ &= U\Sigma V^T - \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T \\ &= C_{xy} - \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T. \end{aligned}$$

The next two directions of maximal covariance will now be given by the second singular vectors $\boldsymbol{u_2}$ and $\boldsymbol{v_2}$ with the value of the covariance given by σ_2. The rest follows the same pattern. The sum of all these deflating terms forms the singular value decomposition of C_{xy}

$$C_{xy} = \sum_{i=1}^{l} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T.$$

Thus, the directions of maximum covariance can be found simply by the singular value decomposition of matrix X^TY.

Least squares using directions that maximise covariance is similar to "Principal component regression" sections least squares. Equation 4.7 can be extended now that we are using the covariance matrix X^TY to define the feature space.

$$U_k^T X^T X U_k B = U_k^T X^T Y = U_k^T U \Sigma V^T = \Sigma_k V_k^T. \tag{4.9}$$

The solution of B is then solved with, for example, a Cholesky decomposition of $U_k^T X^T X U_k$. There is a problem with finding the latent structures if we deflate with SVD components. By doing this we limit the number of components to the number of variables in Y [9]. Lets observe the computation of regression coefficient. The regression coefficient of the case $k = 1$ can be given as bv_1^T, where

$$b = \frac{\sigma_1}{\boldsymbol{u}_1^T X^T X \boldsymbol{u}_1}. \tag{4.10}$$

The approximation of Y will then be

$$bX\boldsymbol{u}_1\boldsymbol{v}_1^T.$$

Hence, the values across the training set of the hidden feature that has been used are given in the vector $X\boldsymbol{u}_1$. So instead of deflating X^TY, we should deflate X by projecting its columns into the space orthogonal to $X\boldsymbol{u}_1$. Using equation 4.1, we can deflate the matrix $X = X_1$ as follows

$$\begin{aligned} X_2 = \left(I - \frac{X_1 \boldsymbol{u}_1 \boldsymbol{u}_1^T X_1^T}{\boldsymbol{u}_1^T X_1^T X_1 \boldsymbol{u}_1}\right) X_1 &= X_1 - \frac{X_1 \boldsymbol{u}_1 \boldsymbol{u}_1^T X_1^T X_1}{\boldsymbol{u}_1^T X_1^T X_1 \boldsymbol{u}_1} \\ &= X_1 \left(I - \frac{\boldsymbol{u}_1 \boldsymbol{u}_1^T X_1^T X_1}{\boldsymbol{u}_1^T X_1^T X_1 \boldsymbol{u}_1}\right). \end{aligned} \tag{4.11}$$

The next direction will be orthogonal to $X\boldsymbol{u}_1$, since it will be linear combination of the columns of the deflated matrix all of which are orthogonal to that vector. The reason for the conjugacy is that $X\boldsymbol{u}_1 = \sigma_1\boldsymbol{v}_1$ is the first eigenvector. Conjugacy means that vectors are in different direction to each other in respect to X if $u^T Xv = 0$. Thus, using the eigenvectors results in features that are automatically conjugate.

Now that we have deflated with $X\boldsymbol{u}_1$ which contributed to the maximal covariance, the maximal covariance of the deflated matrix must be at least as large as σ_2, the second singular value of the original matrix. Also, now that we only deflate matrix X, size restrictions defined by Y do not apply. Matrix Y can be deflated in a similar manner to X but it has no consequences to the subsequent feature extraction. This is usually done for analysis purposes.

There is also a problem with matrix U not relating to the original matrix X directly because the column vector of matrix U are defined from deflated matrices. Because of this we must observe the properties of deflation.

Even though vectors $\boldsymbol{u}_i$ are conjugate they are also orthogonal. Suppose for any $i < j$, we have

$$X_j = Z\left(X_i - \frac{X_i\boldsymbol{u}_i\boldsymbol{u}_i^T X_i^T X_i}{\boldsymbol{u}_i^T X_i^T X_i\boldsymbol{u}_i}\right),$$

for some matrix Z. From this follows

$$X_j\boldsymbol{u}_i = Z\left(X_i - \frac{X_i\boldsymbol{u}_i\boldsymbol{u}_i^T X_i^T X_i}{\boldsymbol{u}_i^T X_i^T X_i\boldsymbol{u}_i}\right)\boldsymbol{u}_i = 0. \tag{4.12}$$

Now if we let

$$\boldsymbol{p}_j = \frac{X_j^T X_j\boldsymbol{u}_j}{\boldsymbol{u}_j^T X_j^T X_j\boldsymbol{u}_j},$$

and also $\boldsymbol{u}_i^T\boldsymbol{p}_j = 0$ for $i < j$. The latter is so because

$$\boldsymbol{u}_i^T\boldsymbol{p}_j = \frac{\boldsymbol{u}_i^T X_j^T X_j\boldsymbol{u}_j}{\boldsymbol{u}_j^T X_j^T X_j\boldsymbol{u_j}} = 0, \tag{4.13}$$

which follows from equation 4.12. Clearly $\boldsymbol{u}_j^T\boldsymbol{p}_j = 1$ so the projection of X_j can also be expressed as

$$X_{j+1} = X_j\left(I - \frac{\boldsymbol{u}_j\boldsymbol{u}_j^T X_j^T X_j}{\boldsymbol{u}_j^T X_j^T X_j\boldsymbol{u}_j}\right) = X_j(I - \boldsymbol{u}_j\boldsymbol{p}_j^T). \tag{4.14}$$

Lets expand this

$$X = X_{k+1} + \sum_{j=1}^{k} X_j \boldsymbol{u}_j \boldsymbol{p}_j^T.$$

If we multiply with U

$$XU = X_{k+1}U + \sum_{j=1}^{k} X_j \boldsymbol{u}_j \boldsymbol{p}_j^T U = \hat{X}UP^TU,$$

where $X_{k+1}U = 0$ like in equation 4.12 and matrix $\hat{X}U$ is the term that we use in regression. Now we can relate our original matrix X to the weight matrix U as follows

$$XU(P^TU)^{-1} = \hat{X}U.$$

By virtue of this we can compute the estimates for the regression coefficients as

$$B = U(P^TU)^{-1}C^T, \tag{4.15}$$

where C is a matrix with columns

$$\boldsymbol{c}_j = \frac{Y^T X_j \boldsymbol{u}_j}{\boldsymbol{u}_j^T X_j^T X_j \boldsymbol{u}_j}.$$

Now we can perform regression with PLS. Next we introduce a couple of algorithms to make computing PLS a lot easier.

4.4.1 Power iteration

There is an other method than SVD to compute eigenvectors of the covariance matrix. Iterative computation of singular vectors can be done using the iterative power method. It can also be used to find the largest eigenvalue of a matrix. This provides us a faster method than SVD when dealing with large matrices. Iteration starts with initialising a random vector. This random vector is then multiplied with the matrix. Then the product is normalised. This multiplying and normalising continues until the product of normalisation converges.

$$A^s\boldsymbol{x} = (Z\Lambda Z^T)^s\boldsymbol{x} = Z\Lambda^s Z^T\boldsymbol{x} \approx \boldsymbol{z}_1\lambda_1^s\boldsymbol{z}_1^T\boldsymbol{x}$$

This vector converge to the largest eigenvector provided $\boldsymbol{z}_1^T\boldsymbol{x} \neq 0$. Convergence is guaranteed since, our starting vector $\boldsymbol{x} = c_1\boldsymbol{v}_1 + \ldots + c_n\boldsymbol{v}_n$ has the following form after k iterations

$$A^k\boldsymbol{x} = \lambda_1^k\left[c_1\boldsymbol{v}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\boldsymbol{v}_2 + \ldots + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\boldsymbol{v}_n\right] \approx \lambda_1^k c_1\boldsymbol{v}_1,$$

where the ratios $\left(\frac{\lambda_i}{\lambda_1}\right)^k \to 0$ as $k \to \infty$ if λ_1 is the dominant eigenvalue. The speed of convergence depends on the eigenvalue ratio. [14]

With the iterative power method, we no longer need to compute the singular value decomposition of our covariance matrix, which is less efficient than the iterative method in some cases. This is so because matrix multiplication requires a larger number of operations ($O(n^3)$). But if the power method is used then the number of operations diminishes to $O(n^2)$. With large matrices this is quite important. Somewhat different techniques regarding the use of kernels can also be applied to PLS. Kernel PLS achieves the same results but much faster. For more information about kernel methods, read reference [9].

4.4.2 Non-linear iterative partial least squares

Next we inspect the Non-linear Iterative Partial Least Squares (NIPALS) algorithm which computes the PLS coefficients. The idea is to decompose both the design matrix X and the response matrix Y.

$$X = TP^T \quad Y = HC^T, \tag{4.16}$$

where $\boldsymbol{t} = X\boldsymbol{u}$ and $\boldsymbol{h} = Y\boldsymbol{v}$. Vectors $\boldsymbol{t}$ and $\boldsymbol{h}$ are the column vectors of their respective matrices. Coefficients $\boldsymbol{u}$ and $\boldsymbol{v}$ are the left and right singular vectors of the covariance matrix X^TY SVD. We use vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ as weight vectors for X and Y. Vector $\boldsymbol{t}$ is the score of X and $\boldsymbol{h}$ is the score of Y. Using the NIPALS algorithm, we decompose matrices X and Y. Firstly, we randomly initialise a vector $\boldsymbol{h}$ and then iterate similarly to the iterative

power method

$$
\begin{aligned}
&\boldsymbol{h} := \boldsymbol{y}_j \text{ for some } j \\
&\textit{Loop} \\
&\quad \boldsymbol{p} := X^T\boldsymbol{h}/\|X^T\boldsymbol{h}\| \\
&\quad \boldsymbol{t} := X\boldsymbol{p} \\
&\quad \boldsymbol{c} := Y^T\boldsymbol{t}/\|Y^T\boldsymbol{t}\| \\
&\quad \boldsymbol{h} := Y\boldsymbol{c} \\
&\text{Until } \boldsymbol{t} \text{ stops changing}
\end{aligned}
\tag{4.17}
$$

In the case when the Y block has only one variable, we can set $\boldsymbol{c}$ to 1 and the last two steps of the loop can be omitted.

The above gives the routine for finding the first set of PLS components and loadings $\boldsymbol{p}$ and $\boldsymbol{c}$. The loadings are the original vector space axes fitted onto the new feature space. Because of normalisation, loadings have a maximum length of 1. If one plots the scores and loadings, this fact helps to measure the loadings' significance in the reduced space. For subsequent components and vectors, we have to deflate matrices X and Y. With this we ensure that we find the largest eigenvalue and the eigenvector per iteration after deflating.

$$
\begin{aligned}
X &:= X - \boldsymbol{t}\boldsymbol{p}^T \\
Y &:= Y - \boldsymbol{h}\boldsymbol{c}^T
\end{aligned}
$$

and repeat the same steps. Matrices T, H, P and C are combined from the respective vectors. To get a regression model we perform regression between T and H,

$$H = TB.$$

Now we can solve Y by using the following equation

$$Y = HC^T = TBC^T = XPBC^T.$$

In this way the NIPALS algorithm uses our knowledge of PLS. Lets look 4.17

a bit more closely to see that our iterative method works.

$$\begin{aligned} \boldsymbol{p} &= X^T\boldsymbol{h}/\|X^T\boldsymbol{h}\| \\ &= X^TY\boldsymbol{c}/\|X^TY\boldsymbol{c}\| \\ &= X^TYY^T\boldsymbol{t}/\|X^TYY^T\boldsymbol{t}\| \\ &= X^TYY^TX\boldsymbol{p}/\|X^TYY^TX\boldsymbol{p}\| \\ &= \frac{1}{\lambda}(Y^TX)^T(Y^TX)\boldsymbol{p} \end{aligned} \tag{4.18}$$

In other words, $\boldsymbol{p}$ is an eigenvector of covariance matrix of Y^TX. In fact, 4.18 is exactly the update rule in the iterative power method uses for computing the largest eigenvalue-eigenvector pair for the symmetric X^TYY^TX matrix.

Now we have shown how PLS projects current data into a new space, which is defined by the eigenvalues and eigenvectors. This is very important, since we can now make analysis on the variance of multivariate data. [25]

4.4.3 Model variation

A model built using PLS follows the multidimensional line it defines. But often the modelled process does not behave the same during each run even if the input parameters are the same. This stochastic nature of a process is something we want to simulate. To explore the possibilities, we expect our data to be of a process that has been performed multiple times with the same input and that there is some difference in the output due to stochastic nature of the process. Time or similar variable that measures the duration or steps of process are important factors since process most likely does not behave the same through the whole process or during all the steps. If the process is compressed into a single observation then the duration does not hold the same meaning as before.

Multivariate linear regression equation could be modified to contain a random term F

$$Y = XB + F.$$

Random matrix F adds random value to the model to simulate stochasticity of a process. As to how to specifically form F, we do not propose any exact methods because it is somewhat problematic and varies from case to case. Each case would need a different aspect to go on and the usefulness of doing

this is not always clear. If the shape of data is non-linear, then the random term F might lead to impossible values if linear model is used. This can also happen when using a non-linear model if the model does not fit to the data all that well. The shape of data is quite difficult to observe in multivariate setting so it might be a bit safer to consider non-linear models.

It might be better to cast aside our linear regression model and focus on the minimum and maximum borders. In this case our model would be the random term F which uses the extrema and minima of data variables to determine range in which the stochastic performance happens. All the process iterations fit in between the minimum and maximum borders and as such they should be used in the model. Simply put, pick a random point between borders at any step or time point and fit a line through those random points. Picking a point uses probability distributions of each variable. Simplest case uses uniform distribution where each point in range has an equal chance. Formed line is one possible behaviour model of a process. With this we can create numerous samples of process behaviour from data. One possible problem is the data cloud of multiple process iterations. It might be impossible for certain behaviour to happen at latter steps if previous steps do not enable it to happen. Evaluation of this should be left for experts as this problem requires different methods or deeper knowledge on the process data.

Other option would be to use stochastic models. These models would fit to our problems rather well but validation parameters become rather hard to understand. Obviously, this methodology needs more investigation. We leave that to future works to consider.

4.5 Validation

The performance of a model can be validated with a number of techniques. Bootstrapping, jackknifing and cross-validation are the most common resampling methods used in statistics. Cross-validation seems to fit our needs in assessing prediction validity. We asses validity for each response variable separately. Multiple types of cross-validation exist, but we inspect the k-fold cross-validation method.

Cross-validation is used to validate the predictive performance of a model. This is done via error terms. In the k-fold cross-validation, training set is split into k equally sized subsamples. Each subsample is omitted once one

Figure 4.1: Ten-fold cross-validation

at a time as in Figure 4.1. Model processes test fold and compares results to measured values. The difference between predictions and measured values form our error term. This is done as many times as there are subsamples. After all subsamples have been omitted once, the mean of the results is calculated. This mean represents the error of the model.

PLS models validation parameters are R^2 and Q^2 where the first one measures "goodness of fit" and the latter estimates "goodness of prediction". We calculate these quantities for each response variable separately in the following equations. R^2 measures how well the model fits to the data. This is calculated using the sum of squares. The Total Sum of Squares (proportional to the variance of the data):

$$\mathrm{SS}_{tot} = \sum_{i=1}^{l} (y_i - \bar{y})^2,$$

where $\bar{y}$ is the mean of vector $\boldsymbol{y}$ and y_i is an individual element of that vector. The Sum of Squares of Residuals, also called the residual sum of squares:

$$\mathrm{SS}_{res} = \sum_{i=1}^{l} (y_i - f_i)^2 = \sum_{i=1}^{l} {e_i}^2,$$

where f_i is the predicted value from the fitted model at the same point as y_i. The formula of R^2 is as follows,

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}. \tag{4.19}$$

So if R^2 value is 1 then the model and the measured data would be identical, meaning that the model fits to data perfectly. On the other hand if the value is near to 0 then the model does not fit to data at all.

Q^2 is determined by error terms like in Figure 4.1. Parameter Q^2 is actually a cross-validated R^2, but instead of SS_{res}, the Predictive Error Sum of Squares ($PRESS$) is used. Model is built from the data in training folds and $PRESS$ is calculated from the test folds.

$$\text{PRESS} = \sum_{i=1}^{l} (y_i - f_i)^2,$$

where f_i is the predicted value from the fitted model and y_i is the real value. This way Q^2 formula is

$$Q^2 = 1 - \frac{PRESS}{SS_{tot}}, \quad Q^2 \leq 1. \tag{4.20}$$

Value of 1 means that the model predicted test fold values perfectly. Smaller value decreases models predictive credibility. [10] [23]

There are other beside Q^2 to assess the predictive performance of a model. The Root Mean Squared Error of Prediction ($RMSEP$) is a widely used method to measure difference between data and predicted values. $RMSEP$ is similar to $PRESS$.

$$\text{RMSEP} = \sqrt{\frac{\sum_{i=1}^{l} (y_i - f_i)^2}{n}} \tag{4.21}$$

The closer the value of $RMSEP$ is to 0, the better.

Measures like Q^2 and $RMSEP$ should be interpreted carefully. Even if Q^2 were somewhat high in value it does not necessarily mean that the model predicts perfectly forever. If the new input values are too different from the previous values, one can only hope that the model guesses right. That is, of course, possible if analysed phenomenon has some degree of regularity in it.

For example, some stocks have some regularity in their growth and decline of worth, but some irregularities might cause spikes (anomalies) in value to either direction. These kind of situations are hard for a model to notice if they were not present in the data on which it was built on. In that sense, R^2, Q^2 and $RMSEP$ are only local quantities.

4.5.1 Variable importance in partial least squares projection

The Variable Importance in (PLS) Projection (VIP) is a variable selection method. This way the relevancy of the input variables is measured. The VIP value is namely a weighted sum of squares of the PLS weights, which takes into account the explained variance of each PLS dimension. The VIP of the jth input variable is

$$\mathrm{VIP}_j = \sqrt{\frac{n \sum_{a=1}^{k} R^2(y, t_a)(u_{aj}/\|u_{aj}\|)^2}{\sum_{a=1}^{k} R^2(y, t_a)}},$$

where u_{aj} is the weight of the jth predictor variable in a component a, k is the number of PLS components, n is the number of variables and $R^2(y, t_a)$ is the fraction of the variance in y explained by the new score $t_a = X_a u_a$ of a component a. This fraction of the variance is calculated like in equation 4.19. The higher the VIP score the more relevant the variable is. Usually $VIP_j > 1$ is considered good. We use the VIP scores to see if certain input variables are necessary. [15] [28]

5 Implementation

In this chapter we experiment with PLS on a dataset to see how the prediction model performs and what other properties of PLS we can use. Simple R-code is provided in Appendix A. This chapter uses "pls" package from the reference [4]. All images related to analysis are located in Appendix C.

5.1 Technology process

The technology dataset originates from a certain process/processes from Nokia which are kept secret because of legal reasons. The dataset contains 58 input variables and 9 output variables. The dataset has over one hundred thousand observations, but only a fraction of it is used. This is so because it takes a tremendous amount of time to compute such a massive collection of data. We settle on a few thousand observations for the sake of convenience.

Before we build our model, we must decide how many components we want to use in our model. The number of PLS components can not exceed the number of input variables. Because our case has a large amount of input variables, it is safer to start with many components, so that we see the model improvement clearly. In this case we use 20 components and the results of this are in Figure 7.2. After 6 PLS components the Q^2 reached good values in most output variables. All response variables were over 0.5 and some were even higher. It has been proposed that 0.5 is a passable threshold and over 0.75 is a good value [10]. It seems that our model predicts the data well enough. After 15 PLS components Q^2 does not seem to grow in most variables. Some variables might improve slightly if more components are used. For example, variable Y1 might get a better Q^2 value if more components were used since it seems to increase a bit in Figure 7.2. There is a possibility that slight increase is caused by some statistical irregularity in our data but used training data is somewhat large so it is less likely. Increase happens because the new model component explains some of the variance of Y1. It decreases when unnecessary variation is added that does not explain Y1 but something else. Parameter Q^2 tends to decrease when a large amount of components are used resulting a higher variance in a model [10] [7]. This problem is called overfitting. We use 15 components in prediction, because

almost all output variables reach adequate validation values.

To make the predictive performance of the model more clear, Figure 7.3 displays the differences of the model predictions and the actual values. Values on the diagonal line mean that the predicted and the measured values are the same. Values outside of the diagonal line have some deviation from the actual values. Do note that the model predicts some values to be lower than zero. This is obviously not possible in our case since the lower limit of our variables is equal to zero. Regression can make this kind of results which should be corrected or noted in analysis. In Figure 7.3, we see that the model predicts rather well but that there are some outliers. Reasoning as to whether outliers should be retained or deleted from the data is up to the analyst, since they might be results of a normal behaviour of the system. The majority of variables are rather nicely on the line, but Y4 blows up quite a bit compared to others. The model obviously cannot predict it too well. To see things from a different angle we can use residual plot. In Figure 7.4 the difference between actual and predicted values is presented using residuals. This method lets us to inspect the prediction results in clearer manner. Fundamentally, Figures 7.3 and 7.4 hold the same information. Now we can see better if variables are non-linear. It seems that Y3 is a little bit non-linear since it starts to curve down. Y2 also seems to curve but more data is needed to make that conclusion. Variable Y6 seems to have a form slightly similar to a 3rd degree polynomial. Current model being a linear model works quite well despite slight non-linearities but if we try to predict outside our value range then these non-linearities could mean that the results have a large error to them. Y4 variables heteroscedasticity is seen more clearly in Figure 7.4. Heteroscedasticity means that the data population has different variance in sub-populations. This does not affect models variable relations or linear estimates, but the model variance will be biased. There are only mild heteroscedasticities in some of our results so it is not a huge problem but it does lessen the prediction credibility when predicting large values because there is larger error.

The overall importance of the variable in the projection (VIP) is seen in Figure 7.5. Some variables explain more than others. VIP provides a good tool to measure predictors modelling power in the model. Variables X25 and X54 have a very little variable importance in the model due to near zero values. Their proportion might increase if more components were included, meaning that they would actually explain variation in output variables but this is unlikely because both variables are close to 0 at all times in data and

the largest of the covariances are already modelled. Variables X16, X17 and X35 have the most importance in the model. It is entirely possible that VIP in the model might not mean that a certain variable has a causal relation with the output variable. It might be a coincidence, but should be verified. If we are performing variable selection, then the variables with VIP score over one are considered important.

In Figure 7.6 we see the regression coefficients for variable Y4 as a boxplot. Variables have either positive or negative correlation. If a regression coefficient is near zero it is safe to say that that variable is neutral to the output variable. Input variable X52 has a high coefficient value to Y4 i.e. they are highly correlated. A closer inspection would be needed to realise whether Y4 is entirely dependent on X52. Other variables have tiny coefficient values, which might be caused by the noise in our data or they could possibly have a high impact on Y4, but not in terms of the coefficients of the model. Figure 7.7 displays the estimated regression coefficients for the variable Y6. The relations between input variables and Y6 are more complicated and finding one variable that explains Y6 explicitly becomes impossible. Variables X16 and X35 have a positive relation with Y6 and variable X31 has a negative relation with Y6. These provide a hint as to what affects variable Y6 and how. This should be validated with concrete testing or expert knowledge. With the coefficients we can look closer to the relations between variables and not solely on prediction results.

The predictive performance of the model was measured with an in-sample measures, but as stated before, out-sample prediction can have a very different results. We predicted from one subset of our data. This subset is three times the size of the training set and is from a random spot in our massive data. None of the two sets share observations. Figure 7.8 displays the predicted and the measured values of the out-sample prediction. As from the Figure 7.8, the new test dataset behaves nicely. Most variables are on the line or close to it, meaning that the prediction is accurate. Only Y4 disperses quite a bit like in the in-sample prediction of the model. Other variables have similar results like in Figure 7.3. It seems that the training dataset contains the overall behaviour of the data. Similar results can be observed from the residual plot in Figure 7.9. Upon closer inspection, variable Y5 disperses more than before putting it into same situation with Y4. It also seems that the model gets higher values than the measured values when predicting larger values of Y7, Y8 and Y9. If we

were to predict higher values then it would be reasonable to assume that they have the same or larger error than what current pictures show.

In Figure 7.10, we have calculated Q^2 with the old model and the new data. The new data is the same we used in the out-sample prediction. Our model still achieved quite good validation values for all output variables. The behaviour with a lower number of components was a bit more erratic to what we have in Figure 7.2, but it does not really matter since a clear improvement can be seen as the number of components grows. The model got a better Q^2 for Y2, meaning that our model predicts Y2 even better than the previous results indicated. Other variables have only a mild difference to what we expected. This is actually good since this was expected. If the Q^2 values had dropped then we would have to reconsider our model. Either the data needs some changes or we need to change the number of PLS components. First point means that training dataset should have more data. The latter is somewhat difficult because increasing the number of components would lead to Q^2 of Y9 value decreasing bit. Though, if the drop is very small it will not have a significant difference in our model but it might model something unnecessary like noise.

For a modest comparison, we also built a Principal Component Regression model from the same data. This can be found in Appendix B.

Further inspection of our data requires more metadata of the process. With metadata we might be able to determine more clearly whether the relations between variables are truly as the model estimates. But this is done using our intuition based on the model and the metadata so it is best left for the consideration of experts with knowledge on the subject.

6 Discussion

We used linear PLS model to model our data with good results. Our data was from a rather complex setting so most likely non-linear model would have gotten more impressive results. There is bound to be some kind of non-linear behaviour that linear model cannot model with same accuracy as non-linear model.

The training set used to build the model was quite large. Because of how cross-validation works it was better to use large set to guarantee that each subset contains enough data. More data means that the model gets even better understanding about data behaviour, thus modelling it better.

As said before, Q^2 parameter is a rather fickle parameter. Higher the value, the better models prediction, but this is not always ideal. Higher value could mean that the model starts to adjust to peculiarities in the training data. It is also better to keep the number of components somewhat low because the subsequent components might model anomalies. Lower Q^2 might be better in some cases to understand the general behaviour. That way the model can handle some variation between datasets better. Because of this it might have been better to build several models with different output variable subsets to optimise the number of components for each output variable. For example, one model would have only Y4 as an output variable and other model would have variables Y1, Y2 and Y3.

If we were to change the input values based on the current model parameters to increase some of the output values then the results would most likely have larger error than anticipated. This change affects other output values as well. As such the predictions should be evaluated carefully. Very large changes in the input values should not be done without verifying. If data behaviour is not strictly linear then the future behaviour might change when moving out of the current range of values. This can also happen to non-linear models so some sort of periodic verifying should be done.

PLS model fits to the Predictive Twin concept quite well since the model parameters estimate the information about the physical asset. The runtime validation also benefits from the PLS model when comparing validation parameters to new parameters. Regardless, usefulness of PLS should be evaluated case by case to see if PLS is good enough.

6.1 Other methods

In this section we introduce other methods than PLS which can be used to predict from data. Namely, Bayesian networks, Support-Vector Machine (SVM) and Artificial Neural Networks. There is also variant of PLS that needs further considering and for this reason is introduced here.

A Bayesian network approaches regression with distributions and probability. Bayesian network model is build from data and this data can be continuous, discrete or both. This fact makes the usage more desirable, since not all the data is continuous. Bayesian network observes variables and their conditional dependencies which can be illustrated via directed acyclic graph. This method could be even better than PLS we have researched, but this would require some further studies.[2]

SVM is a machine learning method in the category of supervised learning models, where it is trained with labelled data. Thus, SVM is mainly used on category classification and is noticed to work best for binary classification (only two categories). Regression version of SVM is called Support-Vector Regression (SVR). SVR builds a model from training data so that the defined cost function is minimized. This is so because if divided into categories, there can be infinite possibilities. If cost function happens to be non-linear, kernel functions can be applied to the function. Kernel functions transform data to the higher dimensional feature space to make it possible to perform the linear separation. In fact, kernel functions can be and are used in various methods. SVM has many variants where different methods are combined. One of these variants is Least-Squares Support-Vector Machines (LS-SVM) method. In LS-SVM, solution is found by solving linear equations. Other notable version is Bayesian SVM. This combo tackles problems in SVM with Bayesian methods. [3]

Other significant method is to use Artificial Neural Networks (ANN). These networks are not an algorithm by itself but a framework for many different machine learning algorithms. ANN consists of artificial neurons and few types of layers. Neurons transmit the input value to neurons in the next layer. Neurons receive those values, process them and send forward as an output of that neuron. Last layer is official the output layer which consists of one or more output values depending how this layer is built. Like in classification output can be two percentage values both representing the probability of the class being correct. ANN can be very accurate, but in

high dimensional cases the networks size increases to enormous proportions. Large amount of data is needed to use large networks. The complexity of the networks rises and running this network is more time consuming. But even so ANN is a mighty tool even if it is a blackbox method. [24]

In this paper we used standard PLS algorithms. But there are other versions of PLS, where some problems of PLS have been corrected or different usage has been devised. One version in particular which piqued my interest was Penalized Partial Least Squares with B-spline transformations. This method is capable of non-linear regression and to prevent model from overfitting to the data, penalty is implemented. This matter requires more research if PLS is preferred method. [17]

There are other methods I/we have not mentioned here that can be used for regression. Methods mentioned here are trendy at the moment and are only suggestion for further research.

7 Summary

Digital Twin will be useful for many fields of study and its benefits are being noticed even though the Digital Twin concept was created some time ago. Technology advances are making Digital Twin possible and in that sense Digital Twin is the logical next step in testing. Even though the Digital Twin is showed as a mathematical algorithm since we only considered Predictive Twin in this paper, Digital Twin is actually a Big Data format. It holds many kinds of information from multiple sources which describe the product, process or service.

Predictive Twin predicts the assets behaviour. In our case, we built a model from data using the multivariate method PLS which is capable of prediction in the form of linear regression. Because of the complexity of the process, multivariate method was chosen. Relations between the input and output data were not that simple, so univariate methods were not a feasible solution. The performance of the multivariate model reached satisfactory results in the Implementation chapter. Because of this we predicted the values of the other dataset with good accuracy.

Theory of PLS shows us the projection to the feature space which is defined by the eigenvectors of the covariance matrix X^TY with a solid proof. This is proven with a proper theory which leads us to the covariance based analysis of data. As a downside to analysis is the fact that correlation is not equal to causal relation so straight forward data analysis is not enough. Deeper knowledge on the analysed asset is imperative. With these, true data relations can be seen and the model can be modified based on that.

The PLS model in the Implementation chapter leads us to many informative directions. The correlation structure of the model shows us dependent relations in the form of VIP and model coefficients. Based on them, we can see models behaviour when input values change. This on the other hand helps us to optimise observed assets behaviour.

Other methods introduced in the Discussion chapter could make model with more predictive accuracy, although most of them are "blackbox" methods. They do not reveal anything of the model expect results which do not help us to improve the observed asset. But it is obvious that each method has something over the other and choosing a certain method should be considered with the problem at hand. In fact, it might be more fruitful to rely on multiple tools than just one.

Relating to research questions stated in the beginning, we have shown how our chosen method works and how it can be used. Additionally, we have provided other intriguing methods which can predict data. Simulating stochasticity of data remains vague, but some ideas came to light. Because of its difficulty it is left for latter works to research and to solve. In this work we did not reach any concrete conclusions on how to simulate stochasticity.

PLS model being informative and accurate we can use it to model an asset. Analysis based on that model helps us to comprehend assets behaviour better and we can validate that model as seen in this research. With increasing data volume the model grasps assets behaviour even better and this improvement can be validated. In conclusion, PLS could work as a Predictive Twin and be used in continuous development and runtime validation.

References

[1] Danilo P. Mandic Alexander E. Stott Sithan Kanna and William T. Pike. "An online NIPALS algorithm for partial least squares". In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017.

[2] Irad Ben-Gal. *Encyclopedia of Statistics in Quality & Reliability*. Wiley & Sons, 2007.

[3] Alexander J. Smola Bernhard Scholkopf. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press Cambridge, MA, USA ©2001, 2001.

[4] Ron Wehrens Bjørn-Helge Mevik. *Nonlinear Partial Least Squares: An Overview*. [Online; accessed March 18, 2019]. 2018. URL: `https://cran.r-project.org/web/packages/pls/vignettes/pls-manual.pdf`.

[5] Yang W. Lee Diane M. Strong and Richard Y. Wang. "Data Quality in Context". In: *Communications of the ACM* 40.5 (1997), pp. 103–110.

[6] John Vickers Dr. Michael Grieves. "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems". In: *Transdisciplinary Perspectives on Complex Systems*. Springer, 2017, pp. 85–113.

[7] N.M. Faber and R. Rajkó. "How to avoid over-fitting in multivariate calibration—The conventional validation approach and an alternative". In: *Analytica Chimica Acta* 595.1-2 (2007), pp. 98–106.

[8] Göran Granholm. *Katsaus kompleksisten järjestelmien elinkaaren suunnitteluun*. VTT, 2013.

[9] Nello Cristianini John Shawe-Taylor. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[10] E. Johansson L. Eriksson T. Byrne, J. Trygg, and C. Vikström. *Multi- and Megavariate Data Analysis: Basic Principles and Applications*. Umetrics Academy, 2013.

[11] Mårten Gulliksson Marlene Mörtsell. *An overview of some non-linear techniques in Chemometrics*. [Online; accessed March 18, 2019]. 2001. URL: http://miun.diva-portal.org/smash/get/diva2:227164/FULLTEXT01.pdf.

[12] Bernard Marr. *Forbes: What Is Digital Twin Technology - And Why Is It So Important?* [Online; accessed January 3, 2019]. 2007. URL: https://www.forbes.com/sites/bernardmarr/2017/03/06/what-is-digital-twin-technology-and-why-is-it-so-important/#146620c82e2a.

[13] MegaM@rt. *Overview-MegaM@rt2*. [Online; accessed March 20, 2019]. 2018. URL: https://megamart2-ecsel.eu/overview/.

[14] mlwiki.org. *Power Iteration*. [Online; accessed March 20, 2019]. 2017. URL: http://mlwiki.org/index.php/Power_Iteration.

[15] K. Saithanu N. Akarachantachote S. Chadcham. "Cutoff threshold of variable importance in projection for variable selection". In: *International Journal of Pure and Applied Mathematics* 94.3 (2014), pp. 307–322.

[16] Kee Siong Ng. *A Simple Explanation of Partial Least Squares*. [Online; accessed March 11, 2019]. 2013. URL: http://users.cecs.anu.edu.au/~kee/pls.pdf.

[17] Gerhard Tutz Nicole Krämer Anne-Laure Boulesteix. "Penalized Partial Least Squares with Applications to B-Spline Transformations and Functional Data". In: *Chemometrics and Intelligent Laboratory Systems* 94.1 (2008), pp. 60–69.

[18] A. Göktas Ö. Yeniay. "A comparison of partial least squares regression with other prediction method". In: *Hacettepe Journal of Mathematics and Statistics* 31 (2002), pp. 99–111.

[19] Himanshi Rajput. *Loginworks: How Digital Twin is transforming Internet of Things (IoT)?* [Online; accessed January 3, 2019]. 2018. URL: https://www.loginworks.com/blogs/digital-twin-transforming-internet-things-iot/.

[20] SAS Institute Inc. Randall D. Tobias. *An Introduction to Partial Least Squares Regression*. [Online; accessed January 3, 2019]. 2016. URL: https://stats.idre.ucla.edu/wp-content/uploads/2016/02/pls.pdf.

[21] A.C. Rencher. *Methods of Multivariate Analysis*. Wiley & Sons, 2003.

[22] Roman Rosipal. *Nonlinear Partial Least Squares: An Overview*. [Online; accessed March 18, 2019]. 2004. URL: https://pdfs.semanticscholar.org/aa15/11d22324ccb8f117dbc1c5383fa95bc1a51c.pdf.

[23] Pradeep Ray Shahriar Akter John D'Ambra. "An evaluation of PLS based complex models: the roles of power analysis, predictive relevance and GoF index". In: *Proceedings of the 17th Americas Conference on Information Systems (AMCIS)*. AMCIS, 2011.

[24] Dustin Stansbury. *A Gentle Introduction to Artificial Neural Networks*. [Online; accessed March 11, 2019]. 2014. URL: https://theclevermachine.wordpress.com/2014/09/11/a-gentle-introduction-to-artificial-neural-networks/.

[25] statsoft.com. *NIPALS Technical Notes*. [Online; accessed March 11, 2019]. URL: http://documentation.statsoft.com/STATISTICAHelp.aspx?path=mspc/NIPALSTechnicalNotes.

[26] The Lustig Group. *Manifesto for Agile software development*. [Online; accessed January 3, 2019]. 2018. URL: http://www.breakthroughperformance.com/blog/2017/7/6/manifesto-for-agile-software-development.

[27] thedailyomnivore.net. *garbage-in-garbage-out*. [Online; accessed March 24, 2019]. 2015. URL: https://thedailyomnivore.net/2015/12/02/garbage-in-garbage-out/.

[28] Svante Wold. *PLS for multivariate linear modeling. Chemometric methods in molecular design*. Wiley-VCH, 1995.

[29] Arnold L. van den Wollenberg. *Redundancy analysis an alternative for canonical correlation analysis*. Psychometrika - Springer, 1977.

Appendix A

To use PLS in R, one can use 'pls' -package or some other package like 'caret' which have multitude of uses and functions. Here we present a simple R-code for PLS.

```
library(pls)
data_model <- plsr(Y ~ X, ncomp = nc, data = dfTrain,
                   validation = "CV") #PLS model
plot(data_model, plottype = "validation")
plot(data_model, ncomp = newnc, asp = 1, line = TRUE)
ypred <- predict(data_model, dfTest, ncomp=newnc)
```

In the code above we build a PLS model with *nc* components (nc has an integer value). Training dataset dfTrain is used in model building between X and Y matrices. After that the validation plot is drawn. Validation plot is used to determinate the amount of components *newnc* used in the latter parts. To confirm this, models in-sample prediction plot is drawn. This shows the difference between actual and predicted values. Lastly, prediction from the new dataset dfTest is performed with *newnc* components

This covers the basics of 'pls' -package. For further study, please visit CRAN-R website [4]. Other programming languages have similar packages.

Appendix B

In theory, PLS and PCR should reach similar results. Because of how the new feature space is defined PLS achieves better results with a smaller number of model components. To test this we built a PCR model from our data and plotted PCR Q^2 parameter in Figure 7.1. The PCR model performs slightly

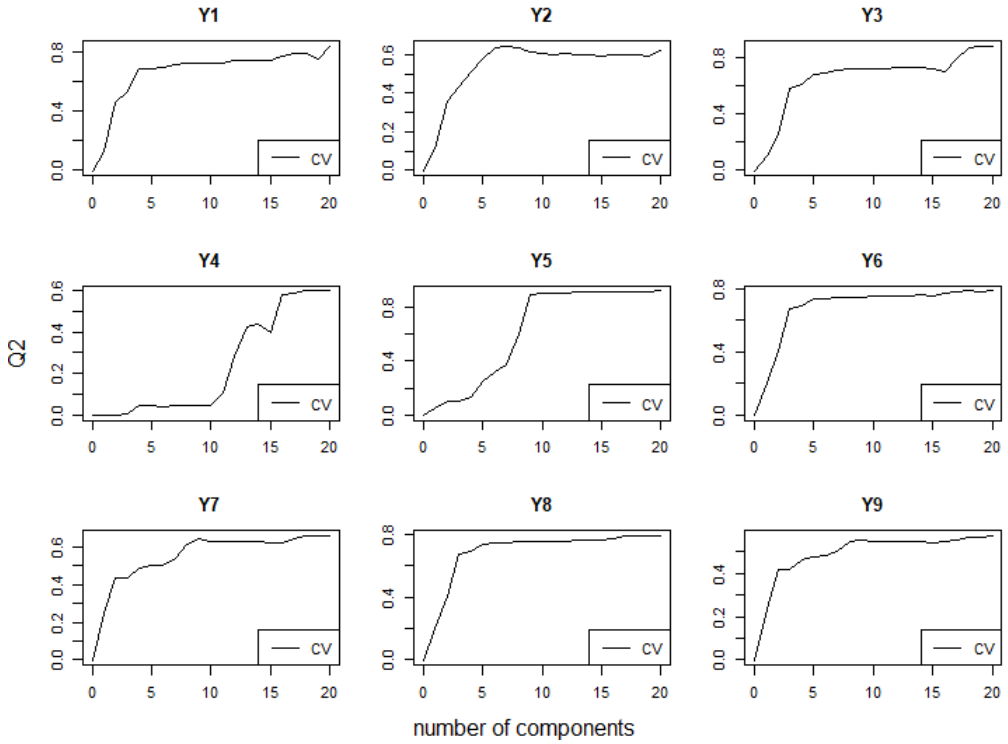

Figure 7.1: PCR models Q^2

worse than PLS model and clearly needs more model components to improve. The PCR model would model noise and other unnecessary things in other variables if we were to reach good Q^2 value for variable Y4. Interpreting the model also gets riskier this way.

Appendix C

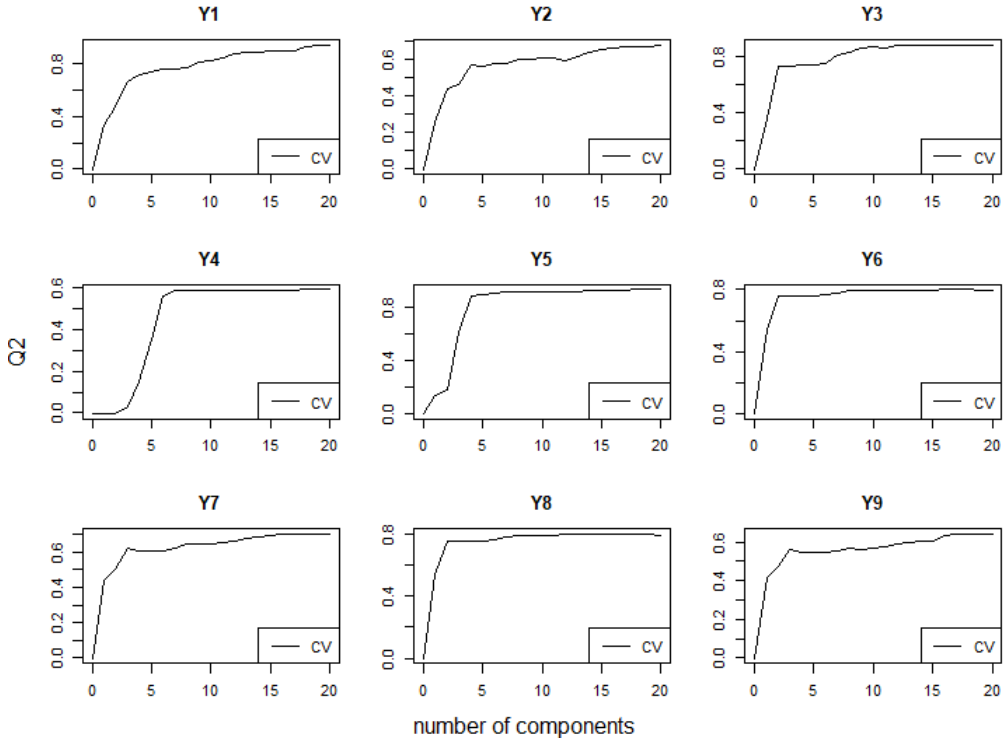

Figure 7.2: PLS models Q^2.

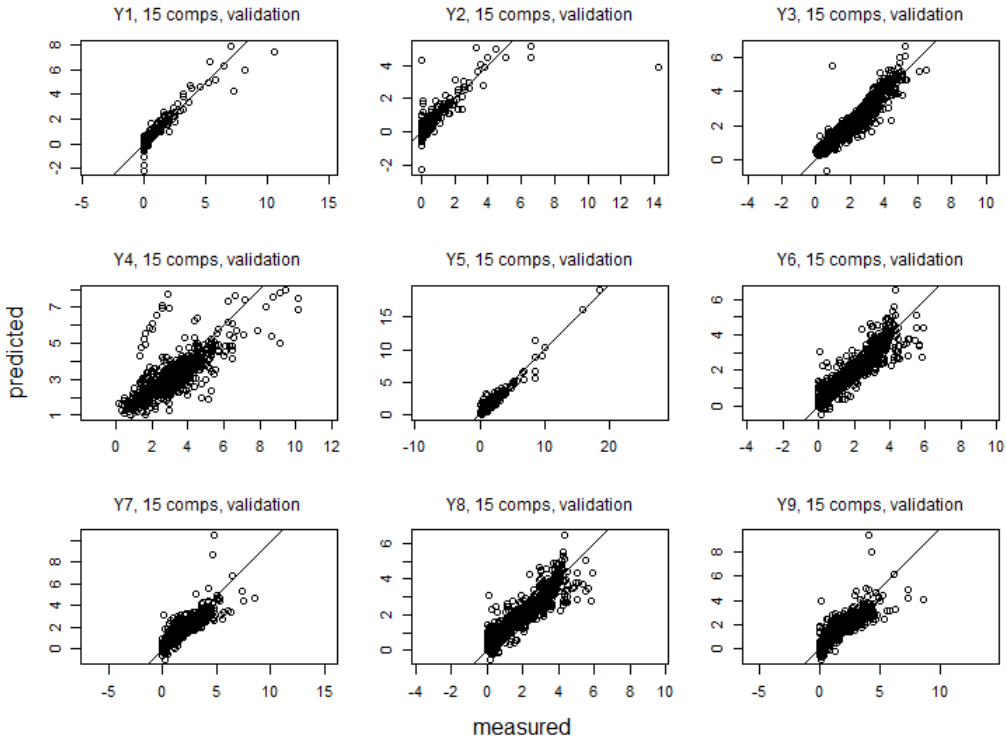

Figure 7.3: Output values of training data vs model estimates.

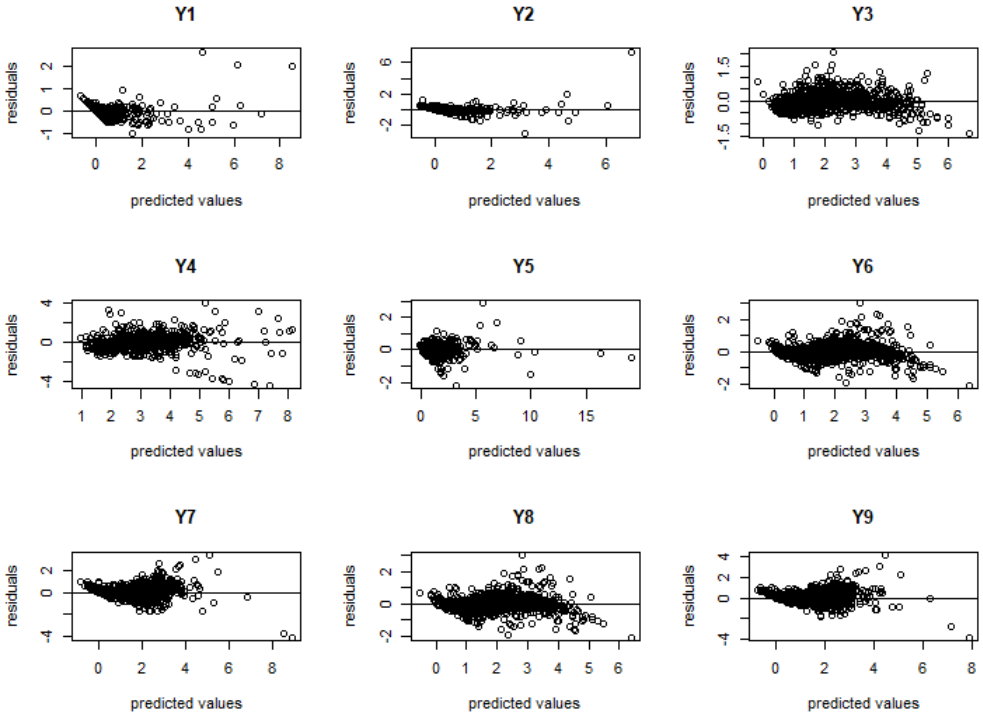

Figure 7.4: Residuals of the PLS model.

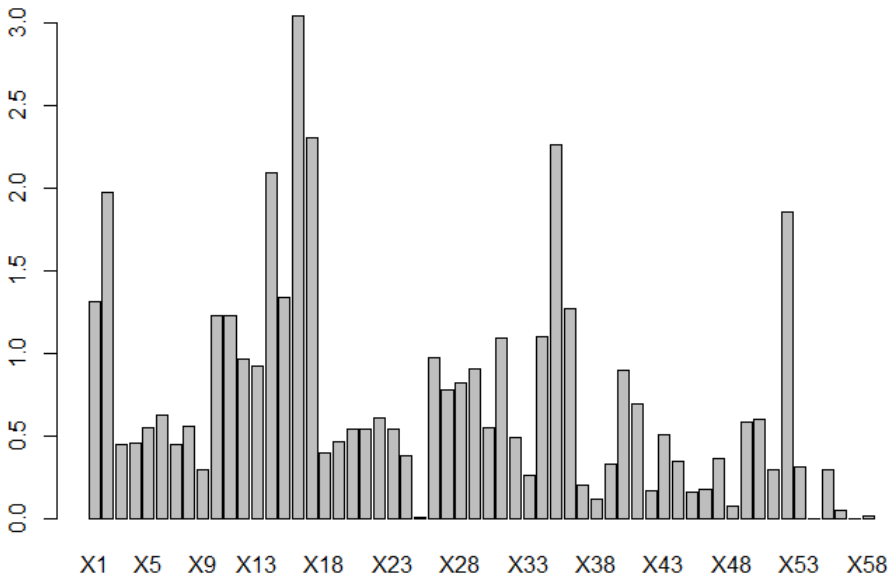

Figure 7.5: PLS models VIP score.

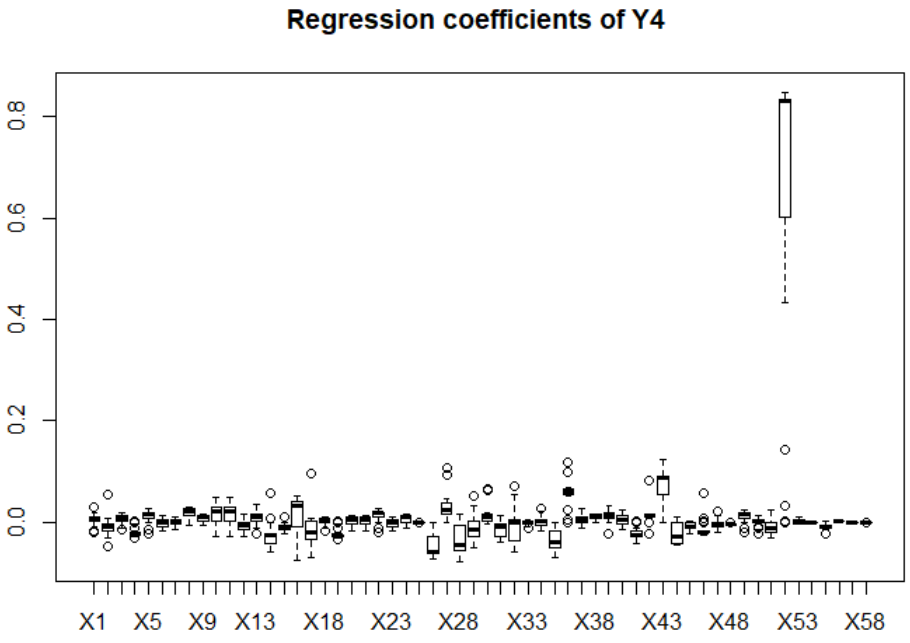

Figure 7.6: PLS models regression coefficients for variable Y4.

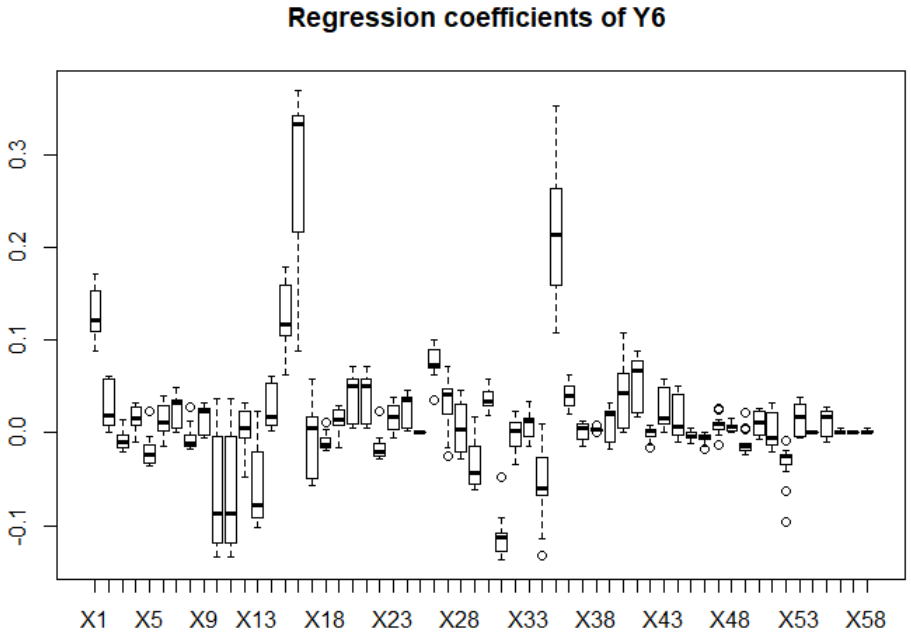

Figure 7.7: PLS models regression coefficients for variable Y6.

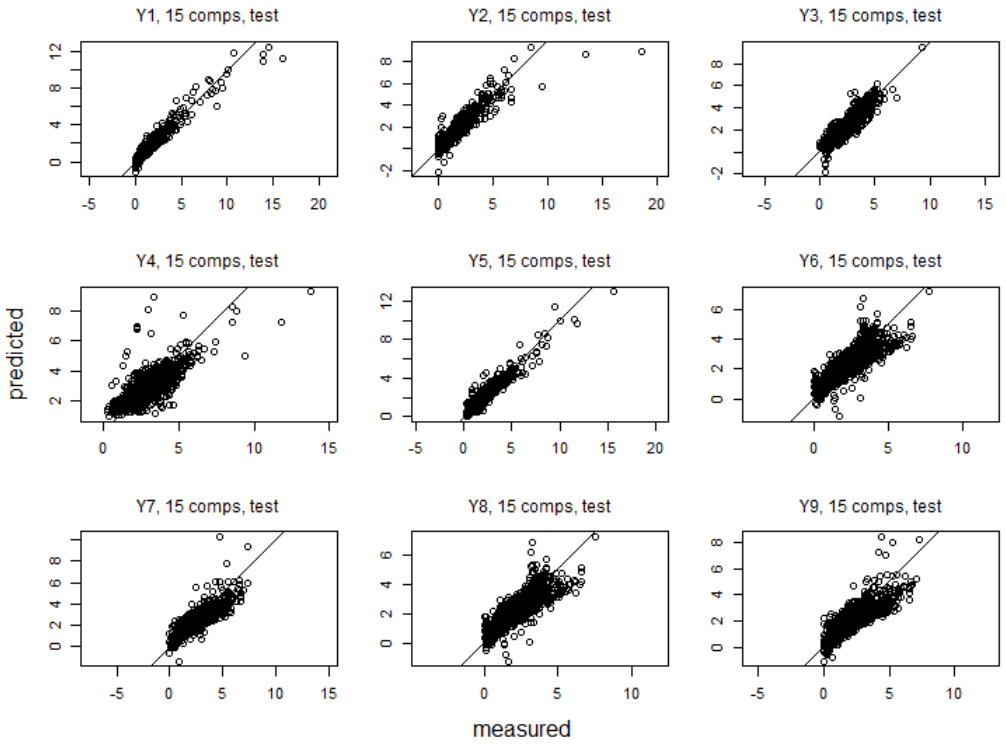

Figure 7.8: Out-sample datasets actual values vs PLS models prediction of out-sample dataset.

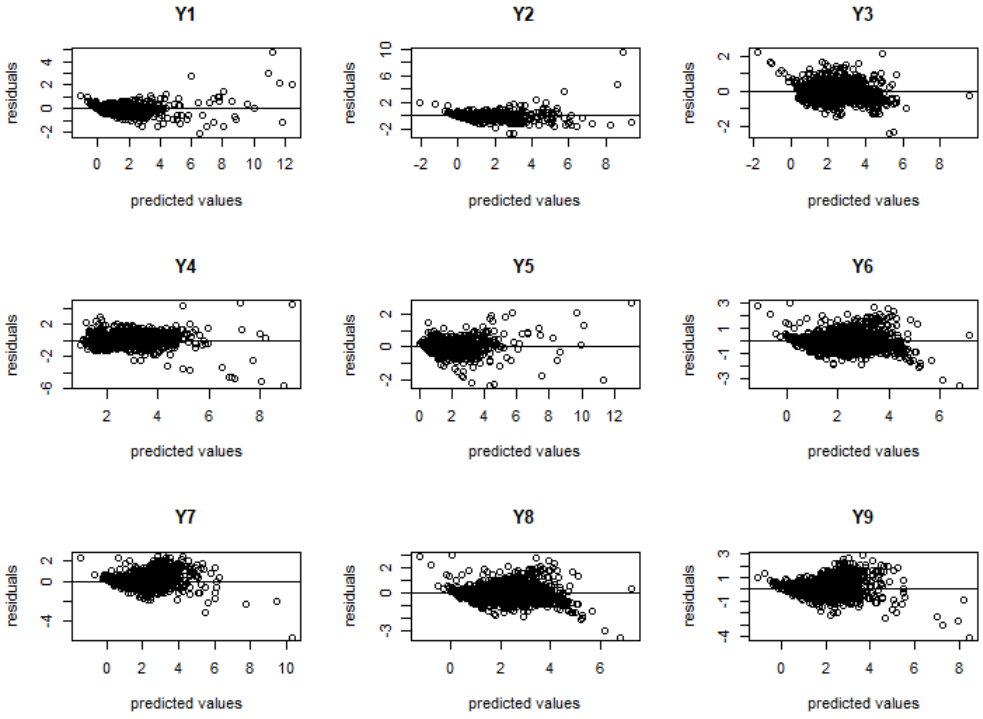

Figure 7.9: PLS models residuals of out-sample prediction.

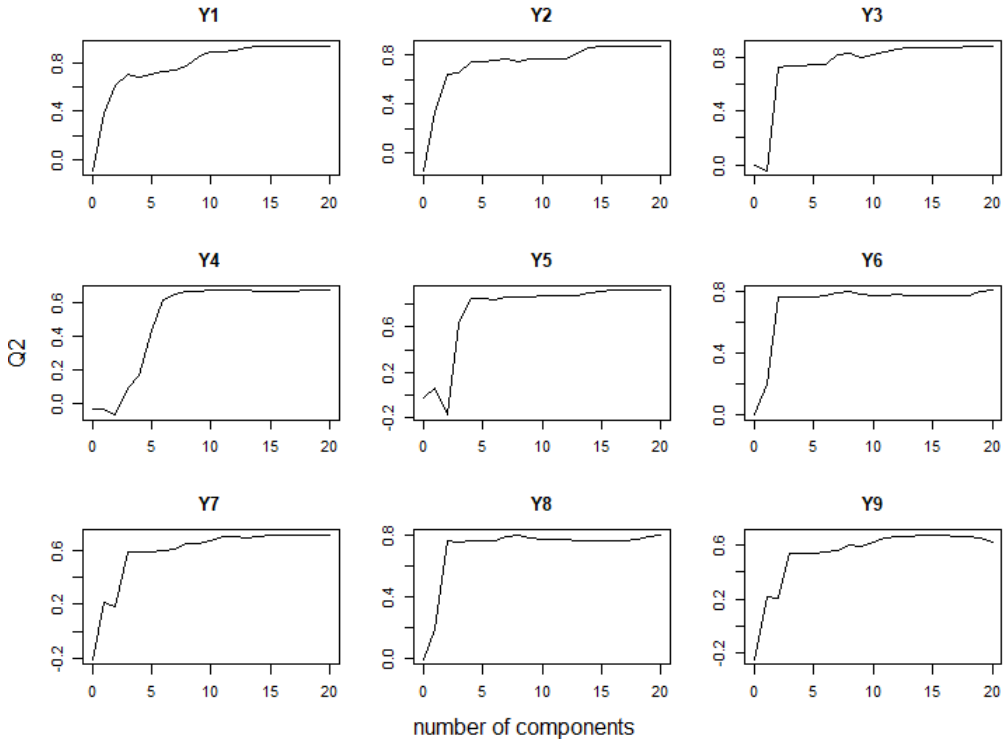

Figure 7.10: PLS models Q^2 of out-sample dataset.